

Single Chip Solution: Implementation of Soft Core Microcontroller Logics in FPGA

Shivam Singh, Prachi Bagave, Soumyadeep Nag

Abstract: This paper proposes a miniaturized model for a particular application by implementing its control circuitry in field programmable gate array (FPGA). This model replaces the conventional circuitry which contains separate unit of microcontroller and complex programmable logic device (CPLD) with a single FPGA chip. Thus it provides a reduced and efficient utilization of resources with increased configurability and flexibility.

Keywords: FPGA, CPLD, miniaturized model

I. Introduction

Hardcore microcontrollers, though being used in various applications, do not provide optimum efficiency because each and every unit is not used in every application. So the unused units thereby create delay, may it be small but countable in crucial application. In real time application, to reduce such delays, there arises a need for application specific integrated circuits (ASIC).

On the similar lines this project aims at configuring application based control circuitry in FPGA. It provides a compact circuit occupying less on board area by implementing the whole logic in a single chip. "System on a chip" technologies enable hardware designers to integrate their board designs, existing of several separate chips, into a single ASIC. [1]. In this way hardware requirements are reduced utilizing logical resources of FPGA. For this transition a pool of adequate high level building blocks favorably implemented as technology-independent VHDL or Verilog description is required. [1] Thus an application specific circuit provides configurability to a great extent.

II. Block Diagram

The project can be modeled as the following block diagram:

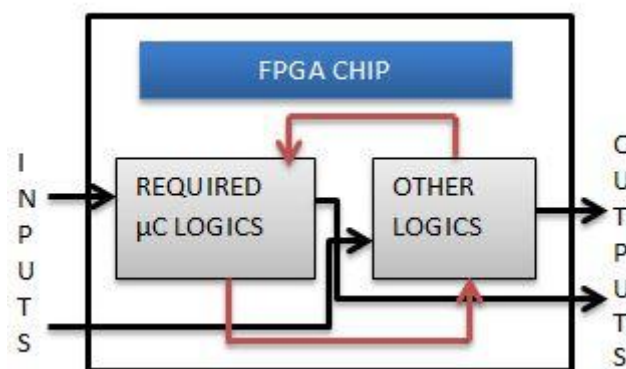


Fig1. BLOCK DIAGRAM

The different units of microcontroller and other logics are replaced by single FPGA. The FPGA is programmed so as to perform the functioning of the hardware's used. Thus the overall functionality is improved due to implementation in a logical way. The logics required in terms of hardware can be included by using a hardware descriptive language. Great amount of flexibility is achieved by reuse of codes.

III. Hardware

FPGA provides a platform with various advantages over other programmable chips in a way that they can be configured by controlling gate arrays at large scale. The large IO blocks configurable ability of FPGA makes them easy to use in any application. FPGAs are made of a variety of matrix-shaped, configurable logic blocks (Complex Logic Block - CLBs) which are arranged on the FPGA in rows and columns. Taken together

as I/O-blocks they form the connections to the logic blocks. Logical operations can be carried out in these blocks.

The application specific this project demands an FPGA with combination of large number of IO's and logical block with quad package. Thus with an approximate estimation of required logical cells XILINX FPGA of Spartan3e series was selected i.e.XC3S500E-4PQ208.

IV. Software Tools

For the implementation of logics a hardware descriptive language (HDL) compatible and prescribed for the device selected by manufacturer is used.

Hence Xilinx 9.2i version is being used for configuring FPGA. It provides a user friendly environment making it easy to learn and understand along this logics can be implemented easily using basic arithmetic and logical operators. This also provides user to program at component level and interlinking to make a complete module that can be verified architecturally using RTL schematic and functionally by waveform simulations. Also, along with the coding facility, floor planning, routing and placement of logical blocks can be generated automatically generated or can be altered manually. Thus this project aims at personalization of the logical blocks as per the user.

V. Approach

The approach used in this project is bottom up approach which allows user to progress in a step by step process. Thus each logical unit can be implemented independently and later be interrelated in an hierarchical way from bottom to top. Further, while implementation of single unit structural approach in VHDL facilitates modular approach to progress to top level.

VI. Implementation

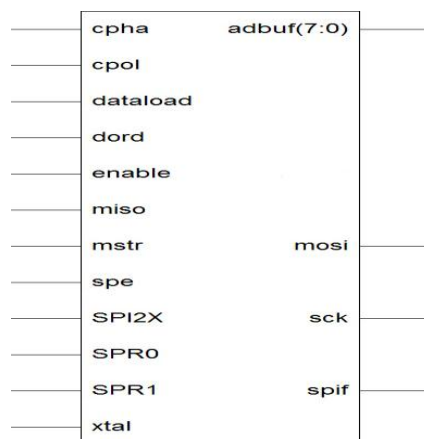
In this project units implemented, for the particular application, includes

- SPI
- TIMERS
- UARTS

A) SPI

The SPI bus is a synchronous serial data link standard that operates in full duplex mode. Devices communicate in master/slave mode where the master device initiates the clock and data frame. Multiple slave devices are allowed with individual slave select (chip select) lines. The SPI bus interface consists four logic signals namely Serial Clock (SCLK), Master Output Slave In (MOSI), Master In Slave out (MISO) and Slave Select (SS).[3]

In master mode, slave works on the frequency of master, and bit by bit data transmission from either side can be achieved. Several other features of SPI like data rotation, clock phase, clock polarity are taken into account and implemented.



.FIG2. RTL SCHEMATIC OF SPI

B) TIMERS

Timers are the basic unit available in any controller to monitor timing related issues. The main principle of working of a timer depends on its registers which count the clock cycles. Timers implemented in this project are 8 bit timers. The timer register is used to count the clock cycles and an interrupt is generated after reaching the

highest possible value. Along with this, output compare register is used to compare with the timer register and create an additional interrupt on compare match.

These timers are used in applications to provide simple delays and to generate various frequencies for different functions.

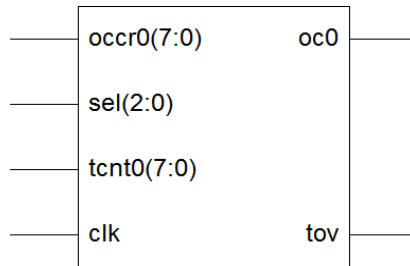


FIG3. RTL SCHEMATIC OF TIMER

C) UART

Universal Asynchronous Receiver Transmitter (UART) is the heart of serial communication. Synchronous communication requires a common clock which should have same phase and frequency for the entire duration of communication. But it is very difficult to maintain same clock frequency and phase for a longer time to all devices which are involved in serial communication. So to avoid this problem, asynchronous communication is used, which operates on different clocks where to check the order of communication extra synchronization bits are added. The internal architecture of UART mainly consists of two modules, transmitter module and receiver module.[2] Along with this the baud rate generator implemented, controls the rate of data transmission and reception.

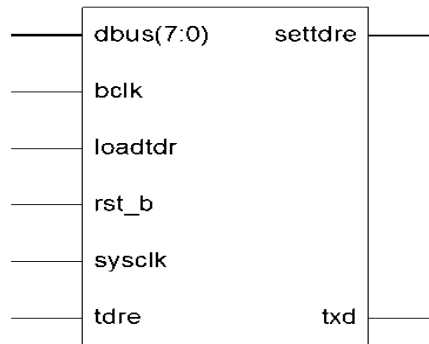


FIG4. RTL SCHEMATIC OF UART

VII. Simulation Results

A) SPI [FIG 6]

Simulation result shown in this paper is of Master mode. A common data register (adbuf) is being shared by both receiving section and transmitting section. So for a given state suppose some data is being stored in data register and when SPI is enabled(SPE high), data is taken into transmitting buffer and sent out serially on MOSI pin along with data received on MISO pin. After 8 clock cycles stored data is transmitted and received data is stored in memory

B) TIMER [FIG 7]

The simulation result shows implementation of 8-bit timer in CTC(clear timer on compare) mode. The 'TOV' interrupt is used to indicate the compare match and 'OC' interrupt toggles on every compare match. The value of TCCR(control register) is set according to the desired mode. The value in the 'TCNT' and 'OCCR' is compared continuously to provide desired delay.

VIII. Conclusions

It could be concluded that separate unit of timers, UART and SPI have been implemented successfully using VHDL. The structural modeling used to implement these logical units provides an effective way in which project can be implicated in step by step way in which provision of debugging at each step is available.

It could also be concluded that flexibility and configurability offered by soft core IP implemented on a single chip provides an effective solution for the short comings of the available application.

References

- [1] Markus Schutti, Markus Pfaff, Richard Hagelauer: "VHDL DESIGN OF EMBEDDED PROCESSOR CORES" in 0-7803-4980-6/98/\$10.00 01~9 98 IEEE.
- [2] Ms. Krupa Joshi, Dipak Patel, Chintan Patel, Rahul Kher: "Design and Simulation of UART IP Core for FPGA Implementation" in National Conference on Recent Trends in Engineering & Technology on 13-14 MAY 2011.
- [3] Veda Patil, Vijay Dahake, Dharmesh Verma³, Elton Pinto: "Implementation of SPI Protocol in FPGA" in International Journal Of Computational Engineering Research (ijceronline.com) Vol. 3 Issue. 1 on JAN'13.

Note: application is copyright of BEL, Pune.

SIMULATION RESULT:

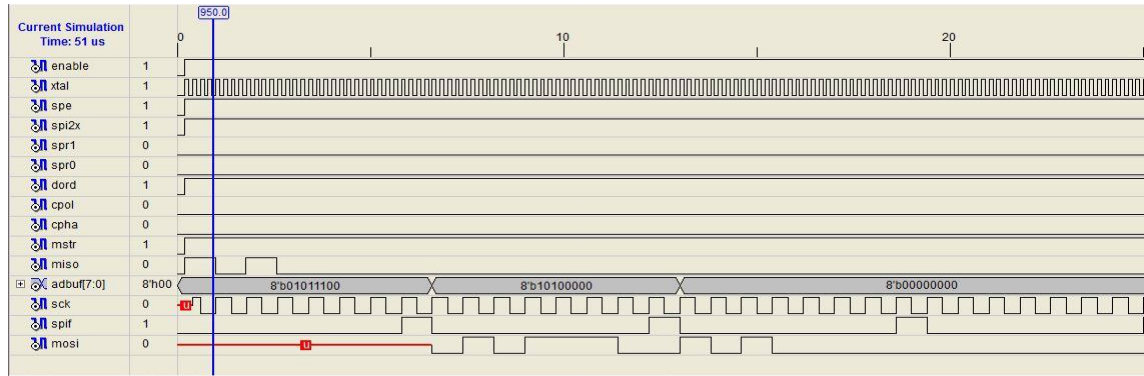


FIG7. SIMULATION RESULT OF SPI

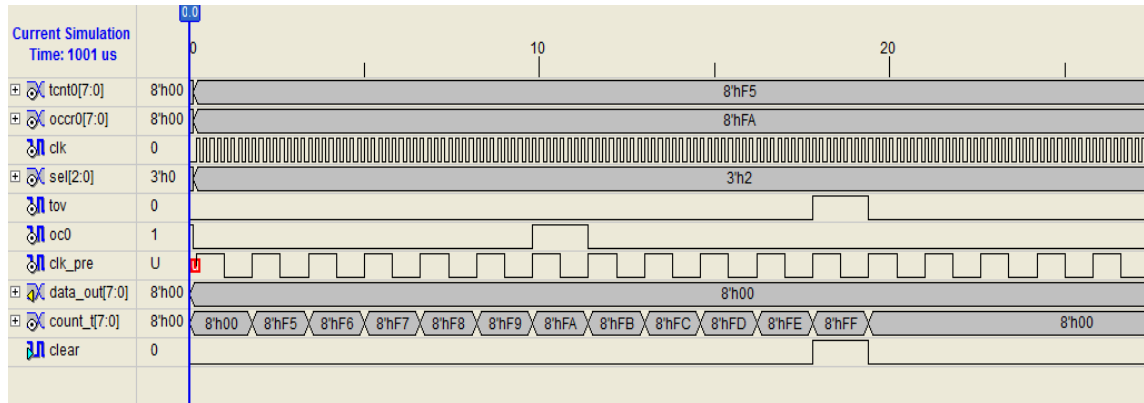


FIG8. SIMULATION RESULT OF TIMER

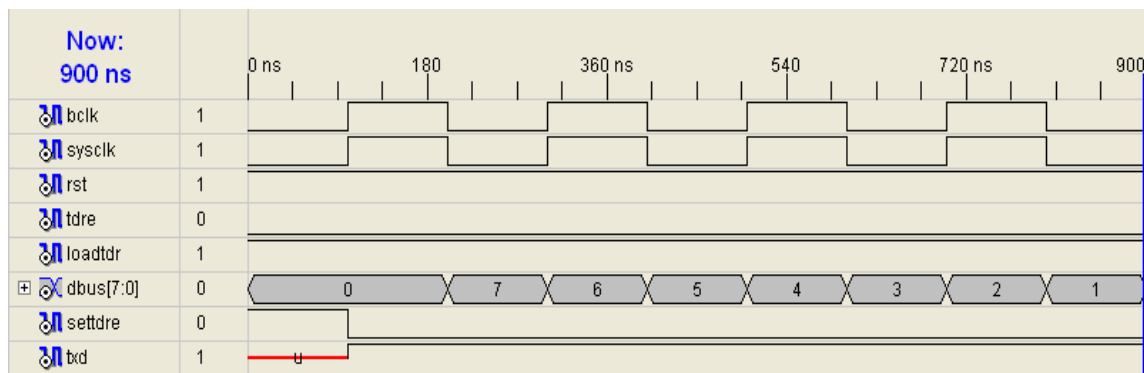


FIG8. SIMULATION RESULT OF UART