

Standalone Command Processor – Onchip Peripheral control and arbitration

Dr. S. R. Ganorkar

Department of Electronics and Telecommunication,
Sinhgad College Of Engineering,
Pune, India
srganorkar.scoe@sinhgad.edu

Amol Bharat Ranadive

Department of Electronics and Telecommunication,
Sinhgad College Of Engineering,
Pune, India
ranadive.amol@gmail.com

Abstract— There are many soft-core processors or microcontroller available in contemporary technological world. Many of those when used for on-chip peripheral control and arbitration are either tend to under use when used for command processors because of their powerful features that are not utilized completely or many of the capabilities and feature become redundant in specific applications. The present architecture of such on-chip command processor is a soft core in its ultra simplified architecture. Its ridiculous simple architecture and yet being powerful enough commands for control encourage it use for many on-chip peripheral controlling purpose. The extended use of the mentions command processor can finds use in debugging & testing a design of a digital circuit that is targeted onto FPGA/VLSI device. The present work discussed is about architecture, commands and typical application scenario. This work also covers the RTL implementation, simulation using test bench in VHDL. Along with simulation result it also covers the synthesis aspects of result for target FPGA from Xilinx.

Keywords- Onchip control, command processor, command interpretation, serial processor

I. INTRODUCTION

Embedded soft processor cores for on chip & off-chip peripheral control are not very common, but at same time there are various architectures that range from wide variety of strong features and large set of instructions/command to a few commands and simple easy-to use architectures. A simple RTL implementation of a command processor with its associated macroinstructions implemented on FPGA device to drive an external peripherals having processor like bus interface [1] and it is effective for external peripherals. Master-serial pair of host (master) and (slave) target-device such that the host sends read or write command to slave and in return the slave sends data packets requested by host or receives data packets from host respectively is efficient and effective approach to interface the on-chip command processor across to user via communication medium like USB [2]. Another such an architecture and instructions are described as an Open On-Chip Debugger (OpenOCD) that aims to provide debugging, in-system programming and boundary-scan testing for embedded target devices [3]. And an interesting discussion that deals with debugging of the Infineon Peripheral Control Processor, PCP for short, within the iSYSTEM winIDEA environment. PCP features and debugging methods are device implementation dependent from Infineon [4]. Command processor with its associated macroinstructions drive an external peripherals but for automotive applications are key area in contemporary world where the processor is master and external peripherals are controlled with simple controller that can be treated as slaves interfaced to the master processor[5].

The present work of a command processor aims and highlights a simplest and east to use architecture with strong emphasis on ease of integration into target VLSI/FPGA application. The architecture is extremely simple yet strong enough tool to have control over internal peripherals and arbitration upon user commands. The RTL implementation in VHDL is targeted to Spartan-3 and Spartan-6 FPGA from Xilinx [6, 7]. The functional verification accommodated using ModelTech ModelSim simulator by developing corresponding test-bench VHDL with the help of advanced tool to build test benches for HDL [8].

II. ARCHITECTURE

Figure-1 shows the architecture of a standalone command processor.

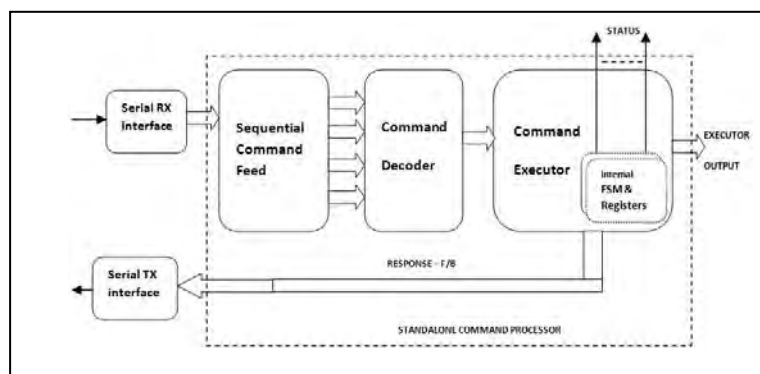


Figure 1. Stanalone Command Processor Architecture

Following Table-1 shows the minimum IOs set that are required for a Standalone Command Processor. This includes the functional description for each of the IO that are available when command processor is seen from top module as an entity.

TABLE I. COMMAND PROCESSOR INPUT-OUTPUTS

Sr. No	Command Processor IO			
	IO Port Name	Direction	Bits	Description/Comments
1.	Reset	IN	1	Asynchronous Active High Reset Input
2.	Clock	IN	1	Clock, maximum clock can be as high as given by synthesis result, minimum should not be less than baud clock of serial interface used
3.	RxRdy	IN	1	Strobe Output from Serial interface
4.	D8_in	IN	8	8 bit command input
5.	NN Out	OUT	8	Numerical Offset Value out can be used to address particular range of memories.
6.	N_Select	OUT	1	This output Sets whenever offset is set by user
7.	Manual/Auto -Mode	OUT	1	Decoded Status Output that indicates the mode of Operation if it is set to Auto Mode (1) or Manual mode (0)
8.	Min-Max Mode	OUT	1	Decoded Status Output that indicates the mode of Operation if it is set to Max Mode (1) or Min mode (0), this will internally affect the offset value by predetermine fashion
9.	Reserved	OUT	1	This status is reserved as an effect to additional set of command. The reserved single status bit can be extended up to more than 50, in effect to support more commands and corresponding control activities

A. Command Fetch/Feed Unit

First, there is always need to fetch a command before it is decoded and executed. In this architecture, the simplicity here that it has command-feed mechanism instead of command fetch. The command fetch mechanism depends heavily on an internal memory counter (which is used as memory pointer/program counter) to fetch the instructions as commands. However, in order to enable the standalone command processor to function as a slave mode, there is provision to feel the command. This command as it enters the internal command feed registers, there is immediate combinatorial decoder acting on the command byte(s). The command feed mechanism is achieved by use of shift register that is bit wide and of 4 location in depth. Thus a 4 X 8 bit serial in is converted to 4 parallel-bytes internally.

B. Command Decoder Unit

Since the command feed mechanism is pure sequential and it hold the command that was fed by user over its, interface the decoding is a combinatorial circuit. This decode action is by achieved use of set of four magnitude comparators. Each magnitude comparator is independent of each other and each work on its own 8-bit of command field. However, in order to incorporate the further decoding of the command that is useful for

the execution module, all these individual fields are also taking into account. Entire functionality of decodes is ANDed to match the symbolic code along with the interpretation of actions that is required over the operand. A set of multiplexers and de-multiplexer are effective in decoding of the commands.

C. Command Executor & Register

Executor unit is an implementation of state-machines which is capable of registering the operand value (byte) in more accurate meaningful way from execution angle. Thus the role of registers in this unit is also to hold the states, and decoded values for precise execution. Status bits are also output of command executor unit. Those status bit indicate the state and corresponding control command that is being activated (Set) or deactivated (Reset). From the above table that generates the offset value NN, it is an outcome from operand of the command field.

D. Timer unit

The timer unit is a hardware module that is implemented as a counter. This counter is enabled when typical set-case for auto-mode is enabled. The generated events of counter are used to change the offset value automatically. One of the obvious predetermined fashion to generate the offset value in an incremented order with the use of additional stage of counter. Integrating or switching over to other methodologies for use of Timer unit to meet the application need should be seamless.

III. COMMAND FORMAT AND DESCRIPTION

Command format is described with the help of the field it has. The format of the command and its fields are given below.

START	CMD	OPRAND	END
-------	-----	--------	-----

Each field is of 8 bits. The command begins with a START field and always ends with END field. The content of these fields are fixed and hexadecimal 53H value is used for START field. This value is for capital S from ASCII table. Similarly the END field is ASCII value for E, which is 45H.

The CMD field for simplicity can be used from ASCII table of Capital letters, Small letters and other symbol including ASCII values representing numbers. Thus from the above top level IOs the CMD field is predetermined as given below.

- Manual Mode : A (ASCII 41Hex)
- Auto Mode : B (ASCII 42Hex)
- Max Mode : C (ASCII 43Hex)
- Min Mode : D (ASCII 44Hex)
- Reserved (Set) : E (ASCII 45Hex)
- Reserved (Reset) : F (ASCII 46Hex)

The CMD is an OPCODE values which can be further extended to more symbols or any value within a limit of 8 bits.

The OPRAND field is of 8 bit wide and hence there is luxury to exercise the set of operands as any value within a limit of 8 bits.

A typical command is as illustrated as follows.

'S' 'A' '0x01' 'E'

From the command illustration above, this is command to set a manual mode and assign the initial offset value for the first range of memory location to process data (process data may include write data into or read from the range that corresponds to the first offset of memory). As seen from the command the S is Start field, A is ASCII value for a command that is internally decoded & executed. The field 0x01 is a hexadecimal value for the operand which is followed by and End field.

On the hardware the command processor was implemented in Xilinx Spartan-6 FPGA. The setup included RS232 interfaced to COM port of PC/Laptop for driving serial commands.

RESULT

The results are from functional and synthesis result targeting for XILINX FPGA. The functional results are achieved by simulation of the HDL model but constructing the test bench for the RTL at each level in the hierarchy of standalone command processor unit. The simulation results are shown from following snapshot of Modelsim simulator in Figure-3.

Figure 2. Simulation Waveforms

The synthesis results show fair are consumption of very few and to the point use of the flip flips and LUTs from the architecture resources of FPGA device. Maximum operating speed is dependent on the critical path for any semiconductor based sequential architecture and hence there is a significant timing optimization achieved that makes the stand alone processor to work as even more than 266MHz of clock frequency as show.

Device Utilization Summary (estimated values)			[-]
Logic Utilization	Used	Available	Utilization
Number of Slice Registers	64	18224	0%
Number of Slice LUTs	69	9112	0%
Number of fully used LUT-FF pairs	49	84	58%
Number of bonded IOBs	54	232	23%
Number of BUFG/BUFGCTRLs	2	16	12%

- Minimum period: 3.755 ns
- Maximum Frequency: 266.326 MHz
- Minimum input arrival time before clock: 3.991 ns
- Maximum output required time after clock: 7.472 ns

The Standalone command processor is ultra simple command processor for on-chip peripheral control and arbitration. The debugging and testing is an extended use for in system hardware debugging and tapping for the values from nets and register states. The approach serves a very cost effective way in the sense from its simple architecture point of view and so also the device resources that are needed for its implementation. Though each command is a set of 4 bytes, and each byte can be driven in as an eight bit parallel input, a sequence of byte as commands needs to be fed serially/sequentially. Hence, commands can be fed through any of suitable

interface as convenient including USB, UART, I2C or SPI interface which cater bit-wide serial i/o. One of a key factor of the command processor architecture carried out here does not have any ALU. This is because Arithmetic and Logical operations are redundant for the control activity in one of the application to which the commands and its processors were designed. ALU integration is left for future scope of this work. In debugging there is need to feed the response & status back to host (master), which can be done through the transmitter unit of same interface which is used to drive commands.



ACKNOWLEDGMENT

I'd like to thank Dr. Ajay D. Jadhav, (P.G. Program Head) and Prof. U. R. More for their insightful comments and constructive suggestions to improve the quality of work.

REFERENCES

- [1] Bhandari, S.U. ; Int. Inst. of Inf. Technol., Pune ; Pujari, S. ; Subbaraman, S., "Embedding Driver for a Peripheral Interface on FPGA", Emerging Trends in Engineering and Technology, 2008. ICETET '08. First International Conference (IEEE), Page: 975 - 978
- [2] Prof. Shashank Pujari, "Embedding Soft processor based USB device driver on FPGA", International Journal of Scientific & Engineering Research Volume 2, Issue 9, September-2011 ISSN 2229-5518
- [3] OpenOCD User's Guide of the Open On-Chip Debugger (OpenOCD) <http://openocd.sourceforge.net/doc/html/index.html> release 0.8.0-dev, dated 9 June 2013
- [4] Infineon TriCore Family On-Chip Emulation, Technical Notes, <http://www.isystem.com>
- [5] Dr. Preeti Bajaj and Dinesh Padole, G.H. Raisoni College of Engineering, Nagpur, India, Design of an Embedded Controller for Some Applications of an Automotives, New Trends and Developments in Automotive System Engineering, Edited by Prof. Marcello Chiaberge, ISBN 978-953-307-517-4, 08, January, 2011
- [6] <http://www.xilinx.com>
- [7] Digilent -Spartan-6 Nexys-3 Reference Manual - <http://www.digilentinc.com/>
- [8] hoTBench.net <http://www.hotbench.net/>

AUTHORS' PROFILE

	<p>S. R. Ganorkar Born on August 6; 1965. He has completed his ME in Adv. Electronics Engineering. His research interests are in Artificial Neural Network and Image Processing. He has 24 years of experience, 13 year in Industrial and 11 years of teaching experience. He is presently working as Associate Professor at E & TC department at Sinhgad College of Engineering, Pune. He has published 12 papers in International journal, 13 papers in International conference and 40 papers in national conference. He is life member of ISTE, New Delhi. He is also a fellow of IETE, New Delhi. email address: srganorkar.scoe@sinhgad.edu</p>
	<p>Amol Bharat Ranadive Amol Bharat Ranadive has completed is BE in Electronics Engineering from Walchand Institute of Technology, under Shivaji University Kolhapur. He has more than 12 years of experience in R&D in VLSI/FPGA designing. He is pursuing Second Year of M.E. in Electronics Engineering in Digital Systems at SCOE, affiliated to University of Pune. Email address: ranadive.amol@gmail.com</p>