FLEXRAY COMMUNICATION CONTROLLER FOR FPGA-BASED AUTOMOTIVE SYSTEMS AS AN IP CORE

[1]P.Satya Shree Sai Ram, [2]Mr.S.Yuvaraj

[1]Student, Department of Electronics and Communication/ VLSI Design
SRM University, Chennai, India
[2]Assistant Professor (O.G), Department of Electronics and Communication
SRM University, Chennai, India

ABSTRACT

Time-triggered protocols like FlexRay have been gaining ground as the standard for high-speed reliable communication in the automotive industry, marking a shift away from the event-triggered medium access used in Controller Area Networks (CAN). This paper describes an FPGA-based communication controller which features configurable extensions to provide functionality that is unavailable with standard implementations. It is implemented and verified on a Xilinx Spartan 6 FPGA, integrated with both a logic-based hardware ECU and a fully-fledged processor-based ECU. We demonstrate that the flexible extensions help enable advanced applications that integrate features like fault-tolerance, timeliness and security.

Keywords- FlexRay protocol, Field programmable gate arrays, communication Networks.

IJAER

INTRODUCTION

In VLSI industry intellectual property core, IP core, or IP block are the reusable unit of logic, cell, or chip layout design that is the intellectual property of one party[1]. IP cores may be licensed to another party or can be owned and used by a single party alone. The term is derived from the licensing of the patent and source code copyright intellectual property rights that subsist in the design.

CAN's popularity [2] in a variety of industries including building automation, medical, and manufacturing. Flex Ray communication controller (CC) which integrates Configurable extensions that augment the CC's capabilities beyond those defined by the Flex Ray standard.

The move towards time-triggered network standards in automotive systems has been driven by the more advanced requirements imposed by advanced mission-critical and comfort features in future vehicles. Widespread event triggered networks like CAN (controller area network) fail to address the requirements of such applications [3]. TT-CAN is an extension of the CAN protocol that enables time-triggered operation by enforcing a slot-based structure, while retaining backwards compatibility with standard CAN. However, TT-CAN suffers from dependability issues and limited bandwidth, and thus it

did not gain widespread adoption. Some research sought to overcome these limitations through hardware extensions on the network controller.

This paper presents highly pipelined FSM based flex relay protocol that consists of a reconfigurable processing module, including reconfigurable compute units and output control logic, CAM memory units, and peripheral circuits.

FLEXRAY PROTOCOL

Flexray is a new communication protocol designed to provide large bunches of data to be exchanged in real-time and with high dependability between electronic control units. It features data-rates up to 10 Mb/s and is accounting for time and event triggered transmissions (beyond CAN). The Flex Ray protocol is developed and standardized by the Flex Ray consortium, and has since been adopted by various automotive companies in production vehicles [6]. These vehicles are complaint with the Flex Ray AUTOSAR Interface Specification Standard [7], which is the industry standard for the software specification of Flex Ray nodes, by which any controller implementation must comply. The fundamental element of the media access scheme in the Flex Ray protocol is the communication cycle, which is repeated overtime is comprised of four segments. Each cycle is comprised of four segments:

- The Static Segment which uses a static slot-based access mechanism and is used to send critical data in a deterministic manner. Any ECU can send a frame of data in the one (or more) slot(s) assigned to it. The slot width is fixed across all nodes on the network.

- The Dynamic Segment which uses a dynamic slot-based access scheme enabling communication of event triggered data of arbitrary length. The slot width is dynamic, depending on the amount of data that needs to be transmitted, and access to the medium is controlled by priorities assigned to the ECUs.

- The Symbol Window which is used to transmit special symbols like the “wake-up” pattern used to wake-up sleeping nodes to initiate communication.

- Network Idle Time which is the idle period used by nodes to make clock adjustments and align and correct the global view of time to maintain synchronization.

IJAER

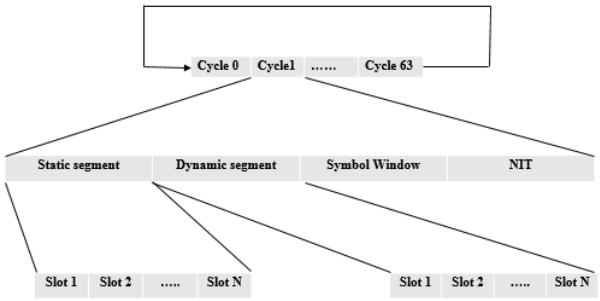


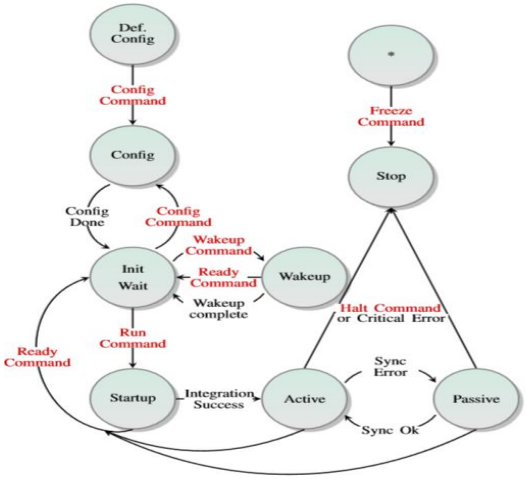
Fig.1 Flexray communication cycle

A. Communication Controller

The FlexRay Communication Controller CC switches between different operating states, based on network conditions and/or host commands, ensuring conditions defined by the Flex Ray protocol are met at all times. The CC architecture, as showninFig.1, comprises the Protocol Engine (PE) which implements the protocol behavior, and the Controller Host Interface (CHI) which interfaces to the host ECU. In the cong state, the host configures the communication controller. Then the host checks wakeup event, based on this decides Modes of operation by state will be next.

The Clock Synchronization (CS) and Medium Interface Layers (MIL) sub modules of the Protocol Engine implement specific functions of the protocol, which are controlled and coordinated by the Protocol Management Module (PMM).

Communication Controller is designed using finite state machine (FSM) methodology. Binary encoded FSM is used which requires lesser flip-flops than the other techniques like one hot, gray.
Speed is independent of number of states, and depends only on the number of transitions into a particular state.



IJAER

Fig 2 Flexray CC Modes of Operation.

The MIL instantiates independent transmit and receive transit buffers to manage temporary storage of Shift Register. This allows us to provide multiple functions with the same set of registers: encode and transmit data as bytes to the HOST.

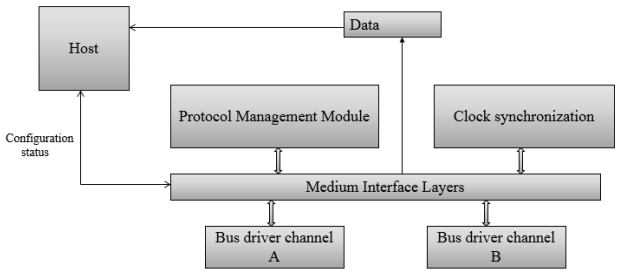


Fig.3 Flexray communication controller

B. Implementation of Onchip processor

Programmable logic need not only be used for application speed-up, it can also be employed as intelligent glue logic for custom interfacing purposes such as in embedded controller applications. Current single-chip embedded processors attempt to provide very flexible interfaces that can be used in a large number of applications. However, they can often result in interfaces that are less efficient than intended. Furthermore, it might be desirable to perform some bit-level data computations in-between the main processor and the actual I/O interface. This paper also investigates the requirements for providing a general purpose field reconfigurable interface for embedded processor applications.

VOLATILE MEMORY

A primary distinction in memory types is volatility. Volatile memories only hold their contents while power is applied to the memory device. As soon as power is removed, the memories close their contents; consequently, volatile memories are unacceptable if data must be retained when the memory is switched off. Examples of volatile memories include static RAM (SRAM), synchronous static RAM (SSRAM), synchronous dynamic RAM (SDRAM), and FPGA on-chip memory.

A. CAM(Content Addressable Memory)

IJAER

Here combined state encodings are stored in the micro coded CAM with parity based content match for both Channel A & channel B. Commands issued by flex CC modes will be matched with CAM and access the HOST through FLEXRELAY interface.

The byte-packed data is written into the receive transit buffers which can then be written into the receive data memory in the host-interface. MIL to ensure two channels are operated parallel and to perform independent operation.

B. *SDRAM*

SDRAM is another type of volatile memory. It is similar to SRAM, except that it is dynamic and must be refreshed periodically to maintain its content. The dynamic memory cells in SDRAM are much smaller than the static memory cells used in SRAM. This difference in size translates into very high-capacity and low-cost memory devices.

C. On-Chip Memory

SRAM devices are more expensive per M Byte than other high-capacity memory types such as SDRAM. They also consume more board space per MB than both SDRAM and FPGA on-chip memory, which consumes none. On-chip memory is the simplest type of memory for use in an FPGA-based embedded system.

The memory is implemented in the FPGA itself; consequently, no external connections are necessary on the circuit board. To implement on-chip memory in your design, simply select On-Chip Memory from the Component Library on the System Contents tab in SOPC Builder. You can then specify the size, width, and type of on-chip memory, as well as special on-chip memory features such as dual-port access.

PERFORMANCE RESULTS

Based on the different commands issued by the controller the Flexray protocol triggers between different modes and functionally verified as shown in fig.4 and the combined state encodings are stored in the micro coded CAM with parity based content match for both Channel A & channel B are shown in Fig.5.

IJAER

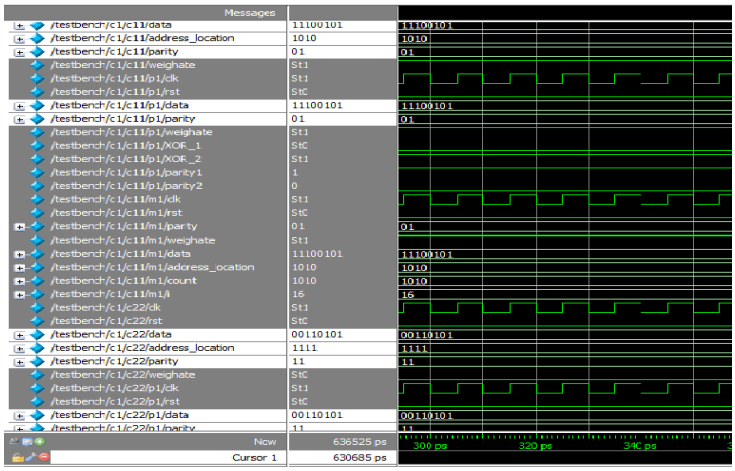


Fig.4 CC modes of operation and its functional verification

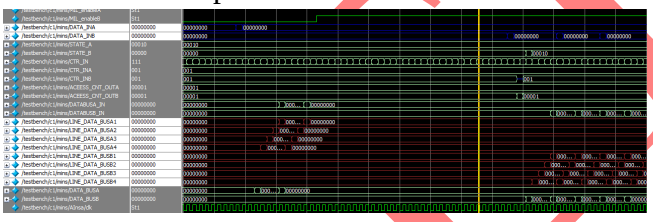


Fig 5. Memory Bank Simulated output

Based on parity the CAM is divided into two parts for reducing the searching time of the data that is stored as state encodings in CAM. The RTL viewer allows you to view a schematic view of the design netlist after analysis and elaboration of netlist before synthesis and fitting optimization. Fig. 6 shows the RTL View of CAM.

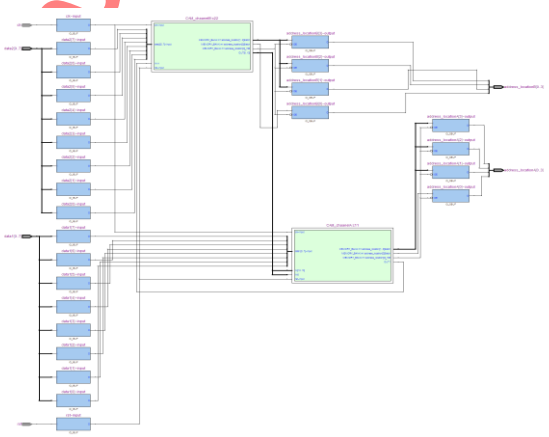


Fig 6. RTL view

IJAER

The comparison of Throughput of the channel A and channel B are computed and compared with and without pipelining shown in the below table.1. The throughput after pipelining at least doubled in both channels. The throughput can be calculated as

"Throughput = no. of bits processed* Fmax"

Parameters	**Without pipelining**	**With pipelining**
Global Clk	3.644GHz	3.648GHz
Channel A	3.9GHz	10.2GHz
Channel B	5.6GHz	11.9GHz

Table 1 Throughput comparison of with and without pipelining.

Using *SOPC Builder* tool the bus-based system consisting of library and user components the Flexray protocol for communication of automotive system was integrated and configured as an IP Core using the library components include processors, memories, bus arbiters and bridges, standard peripherals.

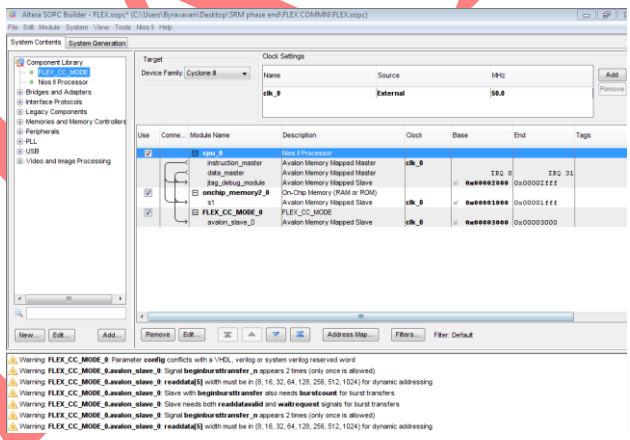


Fig 7. Flexray Protocol as an IP Core.

CONCLUSION

In this paper, we have given an overview of the FlexRay protocol and the generic architecture of the communication controller, as defined by the specification. Advanced computational capabilities like fault tolerance and function consolidation can be built into nodes that integrate complex ECU functions with advanced communication controllers. This approach also improves power consumption compared

IJAER

to the use of discrete controllers. This paper has highlighted the protocol for a new application, which is intra vehicular communication between modules of a car. The design approach for the same and the algorithm is added in this paper. It is recommended to use VHDL tools for the proposed architecture of the application. This will lead to cost effectiveness, simplicity, versatility and other related advantages. The result of interlinking will be presented in future papers.

REFERENCES

[1] Y. Liu and C. Pomalaza-Raez, "A low-complexity algorithm for the on-chip moment computation of binary images," in *Proc. Int. Conf. Mechatron. Autom.*, 2009, pp. 1871–1876.

[2] E. C. Pedrino, O. Morandin, Jr., and V. O. Roda, "Intelligent FPGA based system for shape recognition," in *Proc. 7th Southern Conf. Programmable Logic*, 2011, pp. 197–202.

[3] M. F. Talu and I. Turkoglu, "A novel object recognition method based on improved edge tracing for binary images," in *Proc. Int. Conf. Appl. Inform. Commun. Technol.*, 2009, pp. 1–5.

[4] A. J. Lipton, H. Fujiyoshi, and R. S. Patil, "Moving target classification and tracking from real-time video," in *Proc. Workshop Appl. Comput. Vision*, 1998, pp. 8–14.

[5] J. Kim, J. Park, K. Lee *et al.*, "A portable surveillance camera architecture using one-bit motion detection," *IEEE Trans. Consumer Electron.*, vol. 53, no. 4, pp. 1254–1259, Nov. 2007.

[6] D. J. Dailey, F. W. Cathey, and S. Pumrin, "An algorithm to estimate mean traffic speed using uncalibrated cameras," *IEEE Trans. Intell. Transportation Syst.*, vol. 1, no. 2, pp. 98–107, Jun. 2000.

[7] T. Ikenaga and T. Ogura, "A fully parallel 1-Mb CAM LSI for real-time pixel-parallel image processing," *IEEE J. Solid-State Circuits*, vol. 35, no. 4, pp. 536–544, Apr. 2000.

[8] S. Shreejith, S.A.Fahmy and M.Lukasiewycz, " A Reconfigurable Computing in Next-Generation automotive Networks," IEEE Embedded Systems Letters, vol. 5, No. 1, pp. 12–15, 2013.

IJAER