Design and Implementation of MVB Controller using SOPC Technology

Yongxiang Wang, Lide Wang, Wenqing Liu
Beijing Jiaotong University, Beijing 100044, P.R.China
E-mail: dqwangyx@dr.bjtu.edu.cn

***Abstract*-A MVB (Multifunction Vehicle Bus) controller using SOPC (system-on-a-programmable-chip) technology is presented in this paper. The SOPC technology, developed by Altera Corporation, consists of a FPGA (Field Programmable Gate Array) and a Nios embedded processor. The MVB Access IP (Intellectual Property), which is designed in Verilog HDL (Hardware Description Language), executes the function of data encoding and decoding and checkout. The Nios embedded processor manages the execution of components in the controller. The simulation and experimental result is present.**

I. INTRODUCTION

The MVB (Multifunction Vehicle Bus) is a highly robust real-time field bus specifically designed for control systems built into railroad vehicles. The MVB, together with the WTB (Wire Train Bus) are main part of the TCN (Train Communication Network), which has been standardized by IEC (IEC61375-1) and by the IEEE (Std 1473-1999 IEEE Standard for Communications Protocol Aboard Train). The MVB is a standard communication medium to transport and exchange data among attached devices. These devices, which are physically connected to the bus, may vary in function, size, and performance and at the physical layer level. [1] In order to allow these devices to communicate with each other, a common communication interface card has been designed which is independent of the chosen physical layer and functions associated with each device. The Multifunction Vehicle Bus Controller (MVBC) is an ASIC chip which realizes the physical and major parts of the link layer protocol of the MVB.

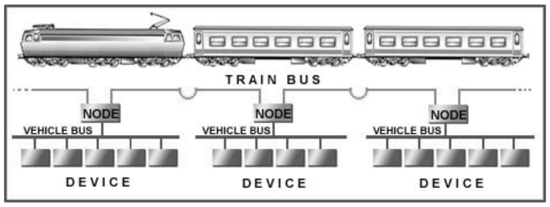

Fig.1. General architecture of TCN in a train

Nowadays, IP (Intellectual Property) designs can be developed and downloaded into FPGA to work with an embedded processor to construct a SOPC environment. [2] Altera's SOPC Builder development tool solves the IP integration problem by placing the integration burden on an EDA tool. SOPC Builder generates all the interconnect logic required to connect the system components in the manner specified by the designer. The use of this design methodology offers significant benefit to the design team through faster integration and hence, increased focus on system development and verification. [3] Due to this advantage, we design a novel MVB interface card under this SOPC environment.

II. SYSTEM DESCRIPTION AND ARCHITECTURE

The hardware structure of a traditional MVB interface card is complicated, and the design will always be annoying. Fig.2 shows the coarse block diagram of a typical application with CPU and MVBC. The MVB card uses a 32-bit RISC processor, such as ARM MCU, as its central processor. A nonvolatile memory is used to store the configuration data. The MVB controller is made by a PLD (programmable logic device). The interaction to the user's application is guided through it, which allows changing of the interface hardware dynamically. Both CPU and MVBC access a common Traffic Store which is the internal arbitration logic, like a large Dual-Port RAM. The Traffic Store contains all volatile data, configuration and status information in an organized structure.

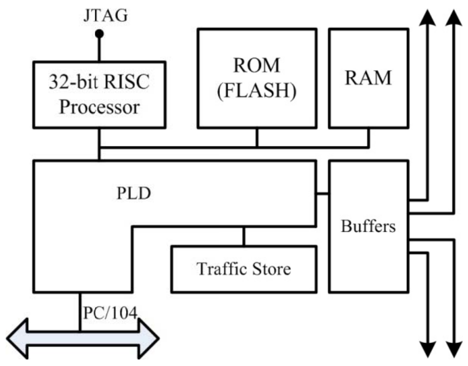

Fig.2. Architecture of a typical MVB card

The size of the Traffic Store can be chosen to be 16K, 32K, 64K, 256K, or 1M bytes where the memory organization (e.g. number of data ports) changes accordingly. Regular static RAMs with arbitration between CPU and MVBC accesses have been preferred over alternative solutions such as an internal Traffic Store in the MVBC or Dual-Port-RAMs. The first alternative was turned down because of increased chip

1-4244-0737-0/07/$20.00 ©2007 IEEE

complexity and cost, and the second alternative was turned down to cost, the limited DPRAM capacity and need of a more complicated mechanism to access internal MVBC registers via the DPRAM interrupt mechanism. [4]

In Fig.2, four digital chips (CPU, MVBC, FLASH, and SRAM) are necessary, and we still need a transceiver to transfer data on different medium, and the power management chip will also be indispensable. It means that we need about six chips to make a typical MVB interface card. With the increase of chip number, more power will be consumed and more disturbances will be generated.

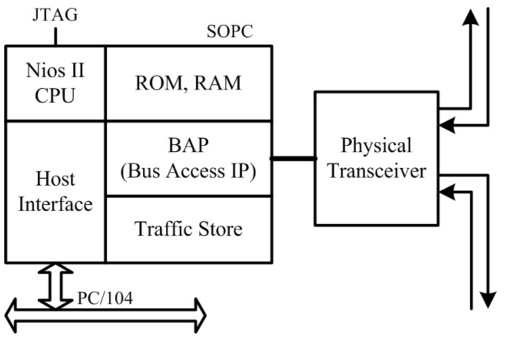

Fig.3. Architecture of a MVB card using SOPC Technology

The proposed design method integrates all the four digital chips (CPU, MVBC, FLASH, and RAM) into one FPGA, and its architecture is shown in Fig.3. The FPGA used in this design is Altera's Cyclone II EP2C8. Cyclone II FPGAs were built from the ground up for low cost and provide a customer-defined feature set for high-volume solutions to cost-sensitive applications. Cyclone II devices are manufactured on 300-mm wafers using TSMC's 90-nm low-k dielectric process to deliver high performance and low power consumption at a cost that rivals that of ASICs. Cyclone II devices support the Nios II embedded processor which allows you to implement custom-fit embedded processing solutions. Cyclone II devices can also expand the peripheral set, memory, I/O, or performance of embedded processors. The EP2C8 provide 8,256 LEs (Logic Elements), 165,888 bits RAM. The 32-bit processor can be implemented in the FPGA by Nios II processor, provided by Altera. The Nios II employ only 1,800 LEs and 18,432 bits RAM, but can provide high processing performance. The ROM, RAM, Traffic Store can be implemented in SOPC Builder. The Bus Access IP, is designed in Hardware Description Language (Verilog HDL), and will be described in the next section. With this FPGA, which contained Nios II, ROM, RAM, Bus Access IP, and a physical-layer transceiver, we can easily make a novel MVB interface card. This card is basically formed by two to four chips in the PCB (Printed Circuit Board). Obviously, the volume and power consumption will be decreased dramatically.

II. BUS ACCESS IP

The Bus Access IP is the core of MVB access processor. The components of this IP can be divided into three parts according to their function, and they are components of physical layer, data link layer, and the interface with application layer. Fig.4 shows the structure of the Bus Access IP.

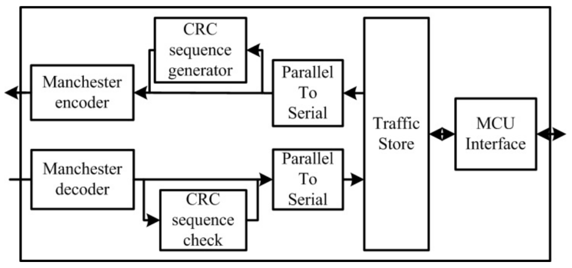

Fig.4. Structure of MVB Access IP

- At physical layer, it implements Manchester encoding and decoding, redundancy processing, bit synchronization.
- At data link layer, it generates function code and master frame and slave frame, media access control (MAC), etc.
- The interface with application layer configures the port and traffic store.

A. MVB frame pattern

The data are Manchester-Biphase-coded and transferred in packets called Frames. These frames are illustrated in Fig.5. Every frame contains a delimiter, 1..16 16-bit data words and at least one CRC. The delimiter distinguishes between Master and Slave Frames. 8-bit CRCs are added to the end of every frame and after every four consecutive 16-bit data words. The combination of Manchester code and CRC guarantee an integer data transfer with Hamming distance of 8. The pair consisting of one Master Frame followed by one Slave Frame constitutes a Telegram.

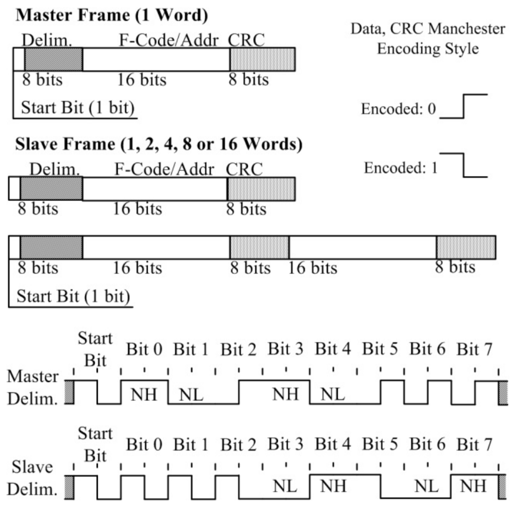

Fig.5. Structure of Master and Slave Frame

B. Encoder and Decoder module

Encoder and Decoder are used to coding bytes stream into Manchester bits stream or recovering bytes from bits stream. The coding style is not as usual Manchester code, because it contains NH (High in a bit time) or NL (Low in a bit time). Normal Manchester encoder and decoder will not work in MVB. In this design, we employed a finite state machine (FSM) to process data coding.

The main function of encoder is transform data from parallel byte to serial bits and Manchester coding. The data to be transferred are stored in encoder buffer firstly by host controller, then begin to be transferred when the encoder received enable signal from host controller. With the clock stepping ahead continually, the state of FSM changes periodically. At each state, the encoder will put out one level appointed by the data in the encoder buffer. When the FSM executed completely, the data transfer ends, and the encoder will wait for the next transfer of frame. The implementation of decoder is also like encoder. The FSM of encoder is shown in Fig.6.

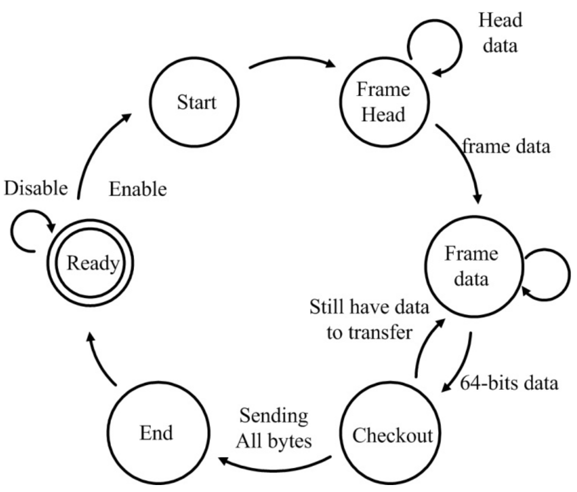

Fig.6. FSM of Encoder

C. Check Sequence Module

The frame data are protected with Check Sequence. For every 64bits data, the check sequence will be calculated by the following expressions.

$$G(x) = x^7 + x^6 + x^5 + x^2 + 1 \qquad (1)$$

Then the result will be extended by even checkout, then the data will be reversed before being sent out. The key implementation code in Verilog is shown in Fig.8.

D. Traffic Store module

The capacity of Traffic Store is determined according to the application. The devices in MVB mostly need 32 or 64 ports, and each occupy not better than 256 bits memory space, so a normal Traffic Store need 8kbits or 16kbits memory space. The proposed design in this paper can provide 64 ports, and each port occupies 256 bits memory. The size of Traffic Store is 2,048 bytes. Altera's Cyclone II EP2C8 contains 20,736 bytes RAM, and it's enough to assign to the Traffic Store. Traffic Store is the interface between the Bus Access IP core and the Nios II CPU, and in this design, Traffic Store is realized through DPRAM (double port RAM). With the help of MegaWizard plug-in Manager in Altera's Quartus II, we can easily create a DPRAM module. Table I shows the configuration of the Traffic Store DPRAM.

```
module caltcrc(rst, sig, pdata, prevcrc, crc);
  input rst, sig;
  input [ 7 : 0] pdata;
  input [7 : 0] prevcrc;
  output [7 : 0] crc;
  reg [6 : 0] buf_crc;
  reg [3 : 0] i;
  assign crc = buf_crc;
  always @(negedge sig or negedge rst)
  begin
    if(~rst)
      buf_crc = 7'h00;
    else
    begin
      buf_crc = prevcrc;
      for(i = 0; i <= 7; i = i + 1)
      begin
        if(buf_crc[6] == 1)
        begin
          buf_crc = buf_crc << 1;
          buf_crc = buf_crc ^ 8'hE5;
        end
        else
          buf_crc = buf_crc << 1;
        if(pdata[7 - i] == 1)
          buf_crc = buf_crc ^ 8'hE5;
        buf_crc = buf_crc & 8'h7F;
      end
    end
  end

endmodule
```

Fig.7. Verilog mode of CRC module

```
module invparity(din, dou);
  input [7 :0 ] din;
  output [7 : 0] dou;
  reg [7 : 0] buf_d;
  reg [3 : 0] i;
  assign dou = ~buf_d;
  always @(din)
  begin
    buf_d = din << 1;
    for(i = 1; i <= 7; i = i + 1)
    begin
      buf_d[0] = buf_d[0] ^ buf_d[i];
    end
  end
endmodule
```

Fig.8. Verilog mode of CRC module

TABLE I

CONFIGURATION OF TRAFFIC STORE DPRAME

Mega Function	RAM: 2-port
Access Mode	With two read/write ports
Word Width	16-bit
Size	2048 words
Clocking Method & Dual clock	Separate 'read' and 'write' clocks

E. Avalon bus interface

This system is composed of several independent components. We use Avalon bus interface to connect them together. Instead of forcing all of the components to fit on a single bus and use the same interface standard, SOPC Builder crates a custom system interconnect that is based on the Avalon interface specification. This specification describes data transfers between a range of different component interfaces and a switch fabric interconnects; this allows any Avalon master to be connected to any Avalon slave by the SOPC Builder tool. [3] Fig.9 shows this Bus Access IP has four Avalon Slave devices and one Avalon Master device.

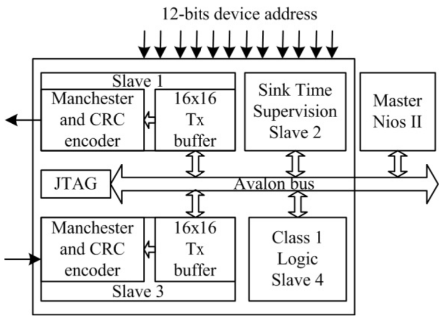

Fig.9. Architecture of Avalon bus system

II. SIMULATION AND EXPERIMENT

Fig.10 and Fig.11 shows the simulation result of Master Frame and Slave Frame simulated in Quartus II. In Fig.10, a 16-bit length Master Frame contained data valued 0x0055 is encoded. The delimiter and checkout value is successfully added. A Slave Frame is shown in Fig.11.

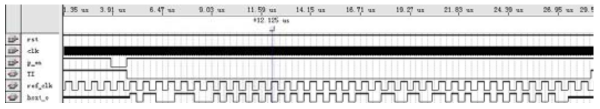

Fig.10. Simulation waveform of Master Frame

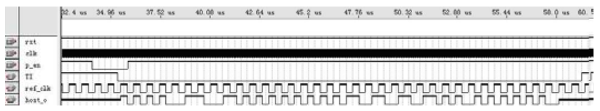

Fig.11. Simulation waveform of Slave Frame

An experimental system included a FPGA experimental board, two monitors with MVB interface and a MVB master is set up. The MVB controller is implemented with a FPGA (Altera Cyclone II EP2C8Q208). A LCD controller is integrated into the MVB controller for the purpose to demonstrate. This card corresponds to class-1 device in MVB network. It uses 2630 logic elements, about one third of the total logic elements in the device. Also, about 8kbytes memory bits, about half of the total memory bits, are occupied.

Fig.12 illustrated the waveform observed on the MVB controller. The experimental result has shown that the proposed design and implementation method can complete MVB communication on one FPGA and a physical transceiver. Now this card is being used in a testing MVB network with one master and several slave devices. It can be configured with 32 ports with each sized up to 256 bits.

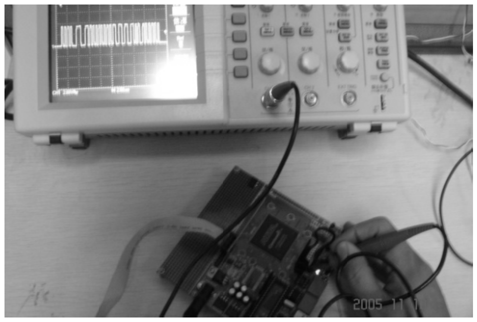

Fig.12. Experiment of the MVB Control IC

II. CONCLUSIONS

The paper proposed a novel MVB controller design method using SOPC technology. The proposed MVB controller was successfully tested at a mixed MVB environment. It combines CPU and MVB controller into one single FPGA, so this method can save PCB space and reduce total power consumption. The availability of multi million-gate FPGA devices and SOPC technology has opened the possibility to create complex single chip systems and can accelerate system performance remarkably.

With the SOPC technology, we have created a novel design and implementation method of the MVB controller. This method can great reduce the chip numbers and card size and power consumption relative to the traditional method. And the design is based on a soft IP core, so it can be easily integrated into other applications.

REFERENCES

[1] International Electrotechnical Commission, “IEC61375-1 Train Communication Network”, 1999

[2] Ying-Shieh Kung, Gua-Shieh Shu, ``Design and implementation of a control IC for vertical articulated robot arm using SOPC technology,'' in IEEE International Conference on Mechatronics, pp. 532-536, 2005

[3] Stefano Zammattio, ``SOPC Builder, a Novel Design Methodology for IP Integration," in Proceedings of International Symposium on System-on-Chip, pp. 37, 2005

[4] Georg Alexander zur Bonsen, ”The Multifunction Vehicle Bus (MVB),'" in Factory Communication Systems, pp.27 - 34, 1995

[5] Delay. B, ``Accelerating system performance using SOPC builder," in Proceedings of International Symposium on System-on-Chip, pp.3 - 5, 2003