Rapid Prototyping of a Communication Controller for the CAN Bus

A. Winter D. Bittruf Y. Tanurhan K. D. Müller-Glaser

Forschungszentrum Informatik Karlsruhe
Dept. of Electronic Systems and Microsystems
FZI/ESM
Haid-und-Neu-Str. 10-14, D-76131 Karlsruhe, Germany
esm@fzi.de

Abstract

This paper presents the rapid prototyping process for a communication controller design. It has been specified at a very high abstraction level with no commitment to the final realization architecture. Via VHDL code generation and synthesis a gate level netlist has been generated. For a fast verification of the specification a hardware emulator and a PC have been configured with the design and have been integrated into a real communications network. By this means it has been possible to discover specification failures as fast as possible and to avoid time consuming redesigns.

1 Motivation

The design of up-to-date, electronic systems for specific applications is more and more influenced by time-to-market demands. As incomplete and inconsistent specifications lead to integration problems which often cause costly redesign cycles when detected very late, they have to be completely verified against system requirements as early as possible [1]. Growing complexity, frequent interaction of reactive systems with their environment and real-time demands make this task more difficult. In addition, the need to minimize design time and costs more and more leads to mixed hardware / software structures on the same chip; up-to-now the suitable techniques and tools for the concurrent design of hardware/software systems are still research topics.

A communication controller for the CAN bus is a typical example for a reactive embedded system. It has to react to input signals both from a host and from the CAN bus in a specific amount of time, following a well-defined protocol. Moreover, the modular structure of the controller's functionality makes it suitable for a mixed hardware/software codesign with the aim to minimize implementation cost, but introducing an additional interface.

In order to focus on functional requirements instead of realization aspects, the controller is specified on a very abstract level which allows first verification runs via simulation. However, simulating the system environment in a testbench is suitable for very early verification purposes only; detailed models are too time consuming and might be incomplete as well. Instead, a CAN network is assembled in which a prototype of the developed controller can be plugged in. The prototype itself consists of a logic emulator for the hardware part and a PC running the software component of the design, connected via the PC EISA bus and protocol.

The following sections present the rapid prototyping process for the communication controller design: section 2 describes the environment beginning with the specification, VHDL generation and synthesis tools to hardware partitioning and emulation. Section 3 provides information about the CAN protocol and functionality. The behavioural specification of the controller is presented in section 4, and section 5 describes the simulation testbench. Section 6 provides information about synthesis and partitioning, and at last the verification of the design in its environment and the results are described in section 7. Future work and conclusion finish this paper.

2 Rapid prototyping environment

The communication controller for the CAN bus is designed in a rapid prototyping system which objective is to overcome the mentioned problems of increasing complexity and incomplete verification. The design is specified at register transfer level in order to obtain synthesizable VHDL code. However, describing a large design in VHDL will result in a very long and hard to read

0-8186-7603-5/96 $5.00 © 1996 IEEE

text file. Especially the hierarchical structure and functional decomposition of a system is much easier described and understood in a graphical illustration. For this purpose Mentor Graphics™ developed the System Design Station (SDS), a graphical front end tool for behavioural specifications which can then be translated to VHDL. The hierarchy of a model is represented in a hierarchy of data flow diagrams. These diagrams are very closely related to the structural VHDL description to be generated, e.g. for each component of the data flow diagram a VHDL entity will be generated. Not only the structure but also the behaviour of the components can be graphically described in SDS. Moore and Mealy type state machines can be modelled as state transition diagrams or state transition matrixes in a very convenient and easy to understand way. These graphical descriptions will be translated into VHDL state machine code without any optimization or encoding. The user also has the possibility to describe the behaviour of a component directly in VHDL. A first verification of the specification can be performed by simulation in a testbench. For this purpose the generated VHDL code can be executed e.g. with the QuickSimII simulator also from Mentor. The stimulation of the simulation is performed directly from the graphical description.

In contrast to tools like ExpressVHDL™ from i-Logix which allow an immediate simulation without generation of VHDL, the specification with SDS provides more flexibility. The experienced VHDL user has the possibility to choose between graphical and textual specification for fine-tuning his model. In addition, simulation with VHDL allows the integration of already existing system components for which no high-level specification model exists.

As mentioned above, verification by simulation is in any way time-consuming, and building a testbench in detail often has to stay incomplete. Therefore a rapid generation of a gate level representation and download to an emulation environment is valuable [3][4]. The generated VHDL code from the specification of the controller is synthesized by Synopsys' Design Compiler™ to a gate level netlist. The target library depends on the chosen realization form: either an ASIC library or Xilinx FPGA primitives.

The gate level netlist is the input to the emulation environment: for the verification of the hardware part Logic Animator™ from Quickturn Design Systems is used, a logic emulator for small and medium size designs up to 50000 gates. This tool consists of a number of reprogrammable components - field programmable gate arrays (FPGAs) - which can be configured with the gate level netlist of a design. Via specific interface modules the emulated design can be easily connected to its system environment and therefore it can be verified extensively and in real-time. In addition this process can be run factors faster than during simulation.

For preparing the programming data of the FPGAs Quickturn offers a specific software tool which automatically converts the gate level netlist to suitable download bitstreams. A specific feature is the overall observability of a design in the emulator. Via software all internal nodes can be provided at the emulator interface modules to the environment. The names of the signals and variables in a design are not changed during the realisation process. Therefore it is easy to identify and access them even after synthesis.

In contrast to traditional prototyping approaches, like building up a system with off-the-shelf components, rapid prototyping by hardware emulation can deal with today's increasing complexity. The download process is just an automatic compilation to a predefined architecture and therefore correct-by-construction.

The process down to the emulator is done in hours, in contrast to days or weeks for the manufacturing of a prototype chip. Moreover, the emulator can be reconfigured and adjusted to new parameters in the same amount of time and can be reused for further designs.

As most embedded systems consist of both hardware and software, a PC connected to the emulator serves for the software prototyping of design's C code. In this way both parts of a codesign can be verified concurrently. Moreover the interface between hardware and software often causes unexpected integration problems [5]. With the mentioned configuration this interface can be easily configured and fine tuned.

Other proposals for validation of system specifications consist of software oriented rapid prototyping environments or "hardware-in-the-loop" approaches. In this case a mircoprocessor or multiprocessor system is used to run C-code generated from a high-level specification. Via specific input/output cards this system under design can communicate with its environment similar to a hardware emulator. There are already systems commercially available like AC-100 or ASCET. However, the response times of such systems are often insufficient (about ms). It is obvious that these systems are restricted to the check of the functionality of the concept, whereas configurable mixed hardware / software platforms which we support are already oriented on the later realization structure.

With our approach different hardware / software partitioning strategies can be tried out in a very short time. And the data of both the hardware and the software of the prototype can be reused for other designs and can serve as a basis for the realization of the product.

3 The controller area network

CAN (Controller Area Network) is a reliable high-performance protocol, which was designed for linking control units in a heavily disturbed environment. It was developed by Robert Bosch GmbH for the automobile industry, where a reliable bus system was needed to reduce the abundant amount of wires in a vehicle. Today, it is used in many other areas, too, where secure communication between control units is needed, e.g. in the automation industry.

The protocol was raised an international standard in ISO-DIS 11898 and ISO-DIS 11519-1. Several chips are already commercially available according to specification 2,0A or 2,0B (Extended CAN). These devices are either microprocessors with integrated CAN interface, that can be directly connected to the CAN bus, or stand-alone CAN controllers, that act as an interface between the host processor and the CAN bus. Another type are the so called SLIOs (Serial Linked Input Output), which are input/output devices with integrated CAN interfaces; these intelligent sensors or actuators provide the means for realisation of distributed systems.

Technical data. Because of the usage in highly disturbed environments, the transmission media should either be an optical fibre or a shielded cable of two wires, where the signal is determined by the voltage difference. The transmission rate can be chosen between 10 kbit/s and 1Mbit/s, with a capacity of 0 to 8 byte of data per message.

Broadcasting. The CAN protocol knows no source or destination address. A message is broadcasted to all nodes simultaneously. Each message contains an identifier of 11 bit or 29 bit (extended CAN) describing its contents. By an acceptance filtering mechanism each node can distinguish the relevant messages from the irrelevant ones.

Multimaster network management. Each node is allowed to start transmission, if the bus is idle (carrier sense). A message on the bus must not be interrupted (non-destructive), unless an error is detected. If two nodes start transmitting simultaneously, the message with the lower value as identifier will have the higher priority, and the other message will be aborted (arbitration without time consumption). Therefore CAN is a CSMA/CA protocol (Carrier Sense Multiple Access/Collision Avoidance). The CAN can be used in real-time environments, as a message with the highest priority will never take longer than 259 bit times for transmission.

Error handling. The following mechanisms are used in the CAN protocol for error detection:

- A transmitter checks whether sent out bits appear on the bus.
- Cyclic redundancy check (CRC)
- Bit stuffing
- Check of the frame formats
- Acknowledge bit

Accordingly up to five simultaneous errors are definitively detected. This corresponds to a hamming distance of 6.

In order to guarantee the systemwide data consistency an incorrect message is destroyed by the sender itself or by a receiving node as soon as the error is detected. The sending node will then repeat the transmission of this frame.

If a permanent failure occurs a node will disconnect itself logically from the network. Thus the functionality of the network is guaranteed, even if certain nodes fail to operate correctly.

With all these qualities the Controller Area Network is very suitable in an environment, where short messages have to be exchanged between peer stations. With a very small overhead secure communication can be established. Additionally the small overhead also results in high baudrates and even real time requirements can be met. For these reasons more and more applications are using the Controller Area Network for communication between electronic control units in a distributed system. [2]

4 Specification at register transfer level

The major part of the bus controller design is done by describing it as a behavioural specification at register transfer level. In this model, the complete functionality of the CAN protocol has to be implemented and still it has to be synthesizable. For this purpose the SDS from Mentor Graphics is used. It will be convenient to reuse parts of this specification in future systems, as it is hierarchically partitioned and the data flow diagrams and state machines serve as a detailed and easy to understand documentation.

The controller is functionally partitioned according to the ISO/OSI reference model for communication systems. The model describes the data link layer, including the logical link control sublayer and the medium access sublayer, and a part of the physical layer, the physical signalling sublayer. The other parts of the physical layer, the physical medium attachment sublayer and the medium dependent interface, are too much dependent on the hardware realization of the network itself to be described at register transfer level. Therefore the physical connection to the CAN medium is realized by an off-the-shelf interface chip. Additionally the fault confinement entity, which is a supervisor for the data link layer, is designed. The higher levels of the ISO/OSI model are not part of the CAN specification and are left to be realized in software.

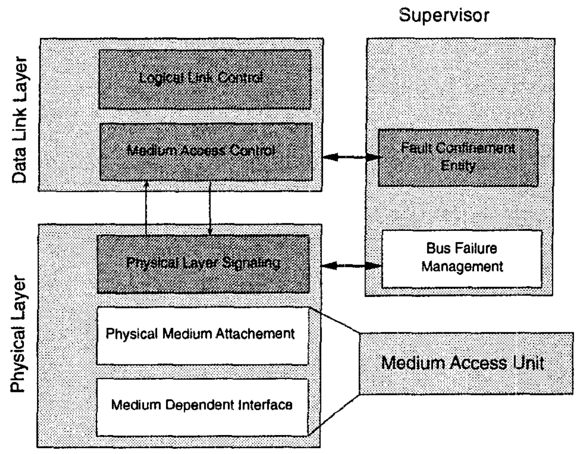

Fig. 1: Block structure of the specification model

Already on this early level a general interface to the client, that wants to use the controller to communicate with other nodes on the CAN, is specified to avoid later integration problems. It is decided to be an eight bit bus and a set of special function registers. So the controller is able to be used to connect a host processor to the CAN.

The main challenge is to design a controller for high baudrates, not by designing a device which is able to be run at high clock frequencies, but by reducing the overhead. According to the CAN specification the length of a bit transmitted by the CAN can be as short as eight clock cycles. Even more it is required, that two clock cycles after sampling the value of a bit the response shall be sent out, e.g. in case the bit had an incorrect value and an error frame has to be sent. Two clock cycles of information processing time is rather short, considering that several situations have to be distinguished and treated differently. Many CAN controllers solve that problem by using an additional internal clock which operates at a higher frequency than the clock used for bit timing purposes. We designed a controller that really needs only two clock cycles of information processing time, and therefore can achieve a transmission rate of 1Mbit/s with an oscillator of 8MHz.

5 The simulation testbench

To validate the behavioural model by simulation a whole CAN environment including a model of the CAN bus itself, a the host processor and several nodes has to be described. This is done in a testbench, where one VHDL unit serves as the host controller for two instances of the designed controller model and therefore can supervise the test. For this purpose the basic structures of the communication software to be used for the host has to be developed. The CAN bus is modelled in such a way, that different kinds of network failures are able to be generated, like disconnected nodes or incorrect values that occur either on the whole network or just at one node.

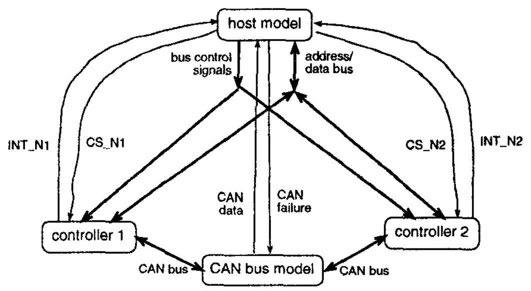

Fig. 2: Simulation testbench

In an even larger testbench, a scenario of four controllers on the network is described. A pseudo random number generator is used to generate frames to be sent and to arbitrarily decide, which node is to send the frame and if an error should occur in transmission or not. With this concept, the controller is validated under all possible timing conditions.

6 Synthesis and partitioning for an implementation using FPGAs

The RTL description is synthesized using the Synopsys™ Design Compiler. This results in a gate level netlist of about 10 000 gates. A netlist in VHDL is generated and simulated using the same testbench as for the validation of the RTL model. The simulation time increases by a factor of 8 compared to the RTL model, though.

As realization technology for the controller 2 Xilinx 4013 FPGAs are selected which seems to be sufficient for 10000 gates. To partition the design and to generate the files for downloading onto the FPGAs the Concept Silicon software from Zycad™ is used. Different partitioning algorithms are tried, but none would end up with less than four devices needed. The hierarchy of the netlist according to the ISO/OSI model is especially unfavourable for netlist partitioning, as a lot of data has to be passed over from one sublayer to the other. So the number of pins per device is the limitation, although there are many unused gates in the devices. All algorithms seems to use the given hierarchy as a starting point. Concept Silicon supports manual partition very effectively though. With the knowledge of the model's structure drawn from the specification a suitable partition for two devices is found manually. This is possible by not grouping the ISO/OSI layers together; instead all the modules related to reception are combined a one group and another group contains all the modules related to transmission. Doing so the data exchange between both groups would be at a minimum.

According to the XACT compilation results the programmed FPGAs should be able to operate at a maximum clock frequency of 5 MHz, so the maximum transmission rate is 625 kbit/s.

7 Hardware/software-coemulation in a controller area network

To verify the system design, the synthesized netlist of the communication controller is downloaded onto a hardware emulator. For this purpose the Logic Animator™ from Quickturn is used, a hardware emulator, that is configured by a PC. In the case specification failures were discovered during emulation, they could be cured by simply manipulating the specification model. Then a redesigned prototype for further verification could be build in a short time using the described rapid prototyping environment.

For verification the emulated controller has to communicate via CAN bus with 2 commercially available chips whereas all components are controlled by a common host.

The host processor for the controllers is implemented by a 486 PC, running specific driver software. This software is developed according to the principles already tested in the simulation testbench and tuned to the host interface of the design. The emulated interface of the controller to be validated is connected via the emulator interface modules to the PC's EISA bus by an extension card especially designed for this purpose. Each driver software module is able to be immediately validated in interaction with the hardware emulator and therefore the interface and protocol (e.g. polling or interrupt controlled communication) between hardware and software and between emulated controller and host respectively are able to be checked easily.

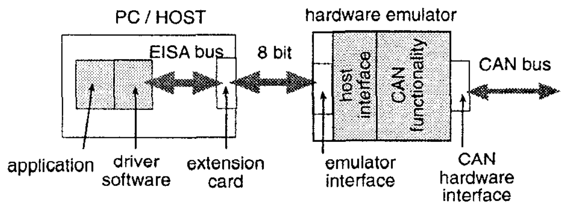

Fig. 3: Block diagram of the design CAN node

For verification the emulated controller has to communicate via CAN bus with 2 commercially available chips, whereas all components are controlled by a common host. In order to use the PC for this task, the commercial controllers are also placed on the mentioned extension board.

This scenario has the same effect as in the testbench: One host processor is able to control the whole network and so an efficient test program is able to be established. Besides verification of the pure transmit and receive functionality, different transmission frequencies and bittiming parameters are checked extensively. In addition specific situations are analysed and induced e.g. for validation of arbitration mechanisms and for generation of defined bus failures in order to check the error treatment mechanisms.

To generate bus failures, the input line of one of the CAN nodes is disconnected from the CAN and directly connected to its own output line. Therefore this node can act as a source of disturbance, as it will not take part in the normal bus arbitration mechanisms. It will then start transmission simultaneously with another node not disturbing the other frame, except for the bit where the failure shall occur. Like this all specified error handling mechanisms can be tested.

Although the specification has been simulated extensively, there are still specification failures detected. This is due to the fact, that the environment is only able to be modelled incompletely in the simulation testbench. The designed controller only communicates with controllers of exactly the same type in the testbench and the model of the host processor cannot represent the exact timing of the EISA bus, for example.

The figure 4 shows the prototyping system. The designed hardware resides inside the emulator which is connected to the CAN bus. PC 1 serves as host processor for the controller and emulates the designed software. It is connected to the hardware emulator via the PC EISA bus. On an extension board inside PC 1 two commercial CAN controllers are found, which are also connected to the CAN bus. PC 2 is needed to configure the hardware emulator and can later be used as a logic analyser to survey the behaviour of the design. The clock frequency used in this network was 4Mhz and the CAN transmission rate was 250kbit/s.

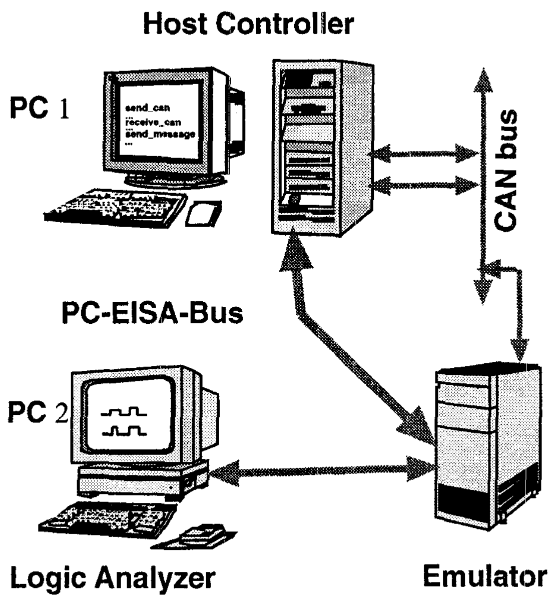

Fig. 4: Prototype of a controller area network

Like this the controller as well as the software is able to be tested under normal operation conditions. Especially the multiplexed data and address bus between the controller and the host processor is not able to be validated sufficiently by simulation only. In addition the controller cannot be observed in communication with known good nodes and therefore misunderstandings of the protocol specification can be excluded.

8 Future work

A main idea in developing this controller and the software was to be able to reuse it in designs of systems which should have a CAN interface. With the VHDL model available it will be easy to reuse it or parts of it, when modelling another system in VHDL. We are planning to design a serial linked input/output unit (SLIO) for a closed loop control system. This unit would be used to survey several sensors and define the values of the actuators. It would be connected to the control unit via the CAN bus. For the design of this system the described rapid prototyping environment will be applied and the developed CAN controller and the software routines will be used as kernels for the SLIO and the control unit.

9 Conclusion

The whole design and validation of the controller chip took less than a year. Most of this time was spent for the analysis of the CAN functionality, requirements definition and the high-level specification. Synthesis, partitioning and download could be performed very fast. For the validation of the design three months were spend; the largest part of time was needed to artificially generate all the error situations for testing the error handling mechanisms.

The experiences which were collected during high level design of the communication controller for the CAN showed the value of the established rapid prototyping system. Especially the hardware/software coemulation turned out a suitable method for verification of reactive systems by observing the system's cooperation with the environment, without having to build an expensive and hard to alter prototype. It is especially useful, if the environment is difficult to model and therefore not accessible to simulation.

10 References

[1] Y. Tanurhan, S. Schmerler, and K. D. Müller-Glaser, "System Level Specification and Simulation for Microsystem Design in the METEOR-Project", *Proceedings of Microsystems Technologies,* Berlin, Germany, 1994.

[2] K. Etschberger: "CAN Controller Area Network", *Hanser-Verlag,* München, Wien, 1994

[3] E. Barke, IMS Hanover: Breadboard zum Simulator und zurück", *slides for a workshop about complex systems verification,* Munich, 1994

[4] D. Behrens, E. Kiel: "Logikemulation mit FPGAs - Der Weg aus der Verifikationskrise?", *GI/ITG Workshop, Anwenderprogrammierbare Schaltungen,* Karlsruhe, 1994

[5] J. Sun, R. Brodersen: "Design of System Interface Modules", *Proc. ICCAD,* Santa Clara, 1992