Teaching System Integration using FPGAs

L. Larsson, A. Klindworth, K. Lagemann, B. Schütz

University of Hamburg
— Dept. of Computer Science / Group TECH —
Vogt-Kölln-Str. 30, D-22527 Hamburg, Germany
e-mail: *<name>@informatik.uni-hamburg.de*

Abstract

To teach students circuit design with respect to system integration in an actual application environment, we use FPGAs as a part of the control system of a technical process. An infrared light based sensor system and an infrared based communication system together with controlling software have been developed by students in this context. All digital processing is implemented with Altera FPGAs that have been designed using AHDL and the MAX+plusII development system supported by EUROCHIP. Implementing the digital components of the technical process with FPGAs gives the advantage that designed components can be tested immediately in-circuit. This is especially useful for teaching where several design iterations have to be expected. Our course showed that most of our students did not realize that constraints emerging from the system environment have to be considered when digital system components are designed and simulated till they made their own practical experience. This paper describes the experiences our students gained during the course.

Introduction

Several chips have been designed at the University of Hamburg during the last years. This work was performed especially in projects and practical courses. Standard cell designs have been preferred. Because of the long time between chip design and delivery, the students interest to integrate the chip in a system typically is much smaller than their interest to design the chips.

Since FPGAs are available in our department, projects are possible which allow the design of chips and the integration into a system environment in one course. In such projects the realization of a complete system is possible, for example the implementation of a technical process. In such a case the circuit design remains important, but it is no longer the main task.

As an example of a technical process we chose a model tramway. The final goal of the project is to control the trams by a "steering by sight" approach avoiding collisions between any of them. This implies that the trams may go one after another coming close up to some millimeters.

A tram system as an example of a technical process

- The task belongs to the area of embedded systems. It is a suitable example for *system integration* and *hardware software codesign* as well.
- A model tramway reduces the effort of making a technical process available itself. This is because the components (rails, switches and trams) needed to realize the technical process are easily available.
- The complexity of the technical process can be modified in a wide range by modifying the railway net topology.
- All trams can be addressed separately via some kind of field bus system, to transmit specific data or to force specific requests.

Steering by sight

One goal of the project is to realize a "steering by sight" for trams going very close one after another. Steering by sight is a sophisticated task, because the velocity and direction of every tram has to be controlled individually – although they use the same electrically connected rails.

For comparison, a railway system – normally using block segment control – could have been more easier implemented than our tram system, but the challenge seemed to us to low.

Central vs. local control

To avoid accidents, the trams control system has to know the positions of trams coming close to each other. In dependence of this knowledge the individual velocities and moving directions have to be controlled. To do this, there are the following two approaches:

- Every tram determines it's current position and velocity and transmits these data continuously to a central computer. This computer acts as a main control unit, which has a current map of all trams. This computer sends commands to the trams in order to control them. This strategy is described in this paper.
- Every tram scans it's environment using a CCD camera or another sensor system. The data taken from the sensor system are processed in real time on board of each tram to control it. This approach is currently investigated by some students working on their bachelor of science and master of science theses.

The approach described in this paper offers several design tasks for the students, with emphasis on the realization of communication channels and protocols (coding, serial/parallel, optical data transmission).

1 Technical Description

The approach shown in figure 1 is one of a lot of possible realizations. The subsystems of the realization are designated for the following tasks:

- A commercial digital multiple train system (Zimo) allows to control the different trams independently from each other. Every tram is addressed by a specific address when sending commands to set a tram's individual velocity or moving direction.
- Those commands can be send by the control computer (a PC) to the Zimo system, which transmits them to the trams.
- At some places within the rails infrared index marks are installed. These index marks send their identification code continuously. A tram passing an index mark receives and stores this code.
- Each tram engine is equipped with a kind of coding wheel. The angular velocity and rotation direction of the engine is determined by two infrared reflex lights focused on it. This approach is similar to mechanical mouses. The two infrared signals control an up/down counter, which gets a reset passing an index mark. This counter delivers a value proportional to the distance to the last passed index mark.
- An additional Zimo command (light switch function) can be used to force transmission of the tram's position, moving direction and index mark identification data – stored in the tram's FPGA – via the infrared link. Put together, this information allows the control computer to calculate the current position of the tram.
- Additionally, each tram contains a reflex light (shown near to the left margin in figure 1) to count the sleepers. This counter can be used alternatively to determine velocities and moving directions as well, instead of the method observing the engine.

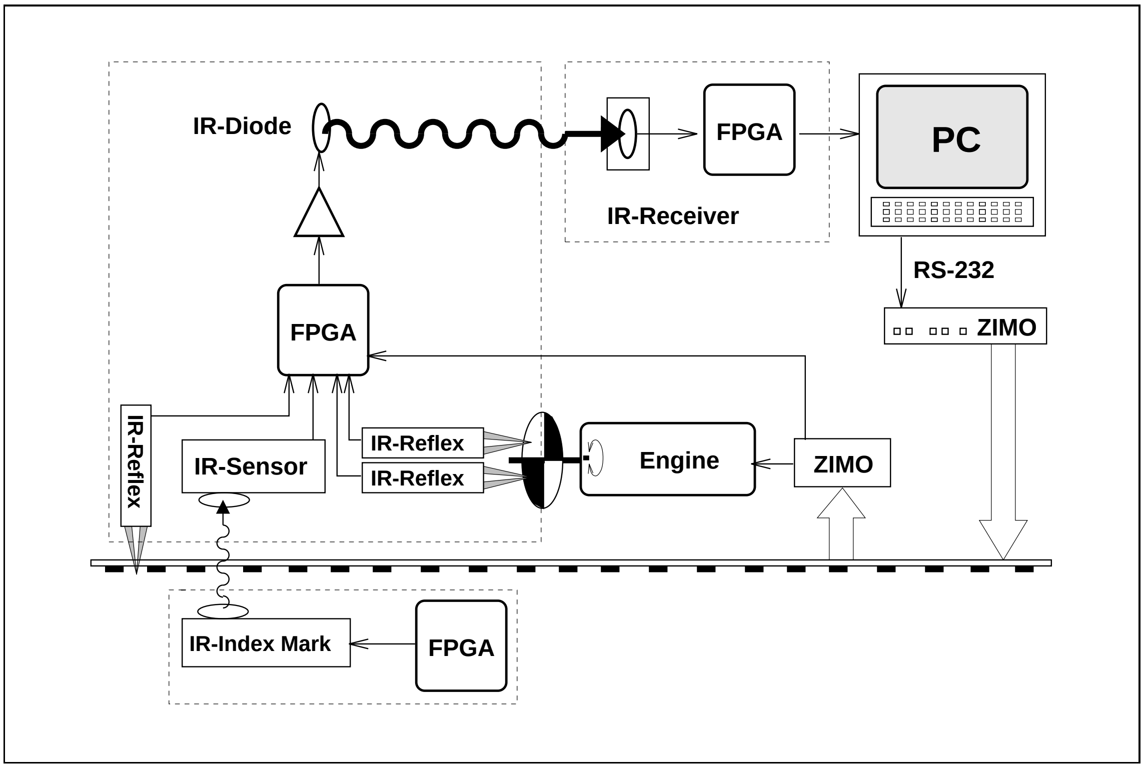

Figure 1: **Main Control Loop.** *One FPGA is installed on every tram. It continuously determines the distance relative to the index mark that has been passed most recently. The current distance and the identification code of the index mark are transmitted on request from the PC via the long range infrared data link. A commercial system (Zimo) is used for electric engine control and to initiate data transmission.*

While this option is not used at the moment, we plan to use this counter to get information about possible errors in engine angular velocity (or errors caused by slip between wheels and rails) and to determine the engine's gear transmission ratio.

Interfacing Infrared Light and Digital Logic

Two kinds of infrared communication facilities are used in our system. Short range infrared links are used for reflex lights and index marks. The range of these links is some millimeters. In this case the infrared LED is driven by a TTL output. As the short range receiver a photo transistor pulls down the input (150 $k\Omega$ pull-up resistor) of a CMOS Schmitt-Trigger inverter (HC14).

The long range infrared link is used to transmit data from the trams to the controlling computer. The transmitter is realized with an infrared LED ($\approx 1A$ pulses) driven by a Power-MOS transistor. A remote control chip (Sharp IS-1460) is used at the receiving end (TTL output). The range of this communication link is some meters. The simple interfaces developed during the course are sketched in figure 4.

Index Marks

The index marks are implemented as short range infrared transmitters located inside the rails. Each mark sends an unique identification code as a continuous bit sequence (fig. 2) that is recognized by each passing by tram. The chosen coding scheme handles the noisy data transmission while approaching the index mark. To attain this feature, the coding scheme allows three different spaces between two edges (start sequence, logical one, logical zero). If all spaces between the edges are according to the coding scheme within a range of acceptance for a whole data word, it is accepted.

Long Range Data Link

Pulse packet ($38\,kHz$ burst) frequency modulation (fig. 3) is used, to transmit logical one and logical zero via the infrared link. The maximum

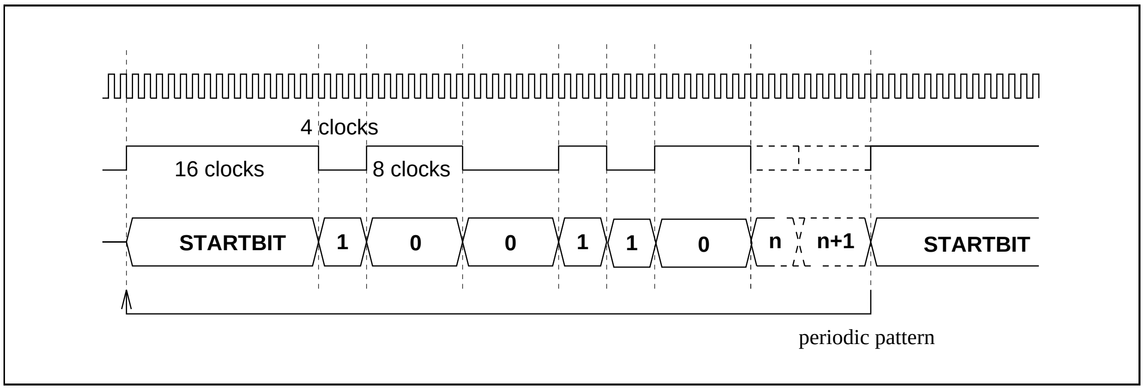

Figure 2: **Index Mark Sequence** *To code the number of the index mark three different pulse lengths in the ratio 1:2:4 are used. The longest pulse defines the start bit and the shortest one defines the logical one. This coding scheme takes the continuous approachment between sender to receiver (short range) into account.*

data rate is restricted by the infrared detector's (IS-1460) response time and latency time. Based on this infrared channel a RS-232 protocol is realized.

Involved FPGAs

Three digital processing tasks (fig. 1) are performed using two different types of FPGAs. The type EPM7128LC84 (128 logic cells, 84 pins, $\approx$ 2500 gate equivalents) is used to realize the on board tram controller. This controller processes the signals from the infrared sensors including index mark recognition. All necessary processing to transmit collected data via the long range infrared link is done by this FPGA, too. Approximately 90% of this FPGA is utilized.

The second FPGA of this type is used for the long distance infrared receiver. This converts the puls packet response signals from the long range infrared detector (IS-1460) to the RS-232 protocol. In addition, the receiver converts the serial data flow into binary words and thus a parallel I/O-card is used inside the controlling computer. For this task approximately 50% of the FPGA is utilized.

Generating the continuous index mark sequence a smaller FPGA of type EPM7032LC44 (32 logic cells, 44 pins, 600 gate equivalents) is used. Its utilization is approximately 50%.

2 Teaching Experiences

In pure digital circuit design, switch level models are valid. These models enfold logical description and delay times. But on system level, different models of system components have to be taken into account. In addition, interactions of different system components are possible. So the digital circuits have to be designed with respect to system level. Simulations have to take these effects into account, too.

Fifteen students participated in the project in the fall semester 1993/94. Three groups were set up. One for determination of positions of trams, one for infrared data transmission and one for programming of the control software. It was surprising that students working close to hardware and microelectronics development were the most active. In contrast to this, the interest in software development was not so strong.

The practical aspects close to the hardware were new experiences for most of the students in the project. Many students recognized for the first time that an additional abstraction level between the logical level and the electrical level has to be taken into account. This means the additional level of carrier techniques (pulse packet frequency modulation, frequency modulation, time length encoding, etc.) to transmit bits with an acceptable noise ratio at fixed bandwidth. This became clear during the development of the modulation scheme of the long range infrared link.

Additionally, the familiar picture of ideal synchronous finite state machines with clearly separated clock and data signals was not applicable in each case. In this context it was new to many students to handle data streams, without explicit clock information. Although the tasks to be solved with hardware seemed simple in principle, the students needed some weeks to design different reliable system components.

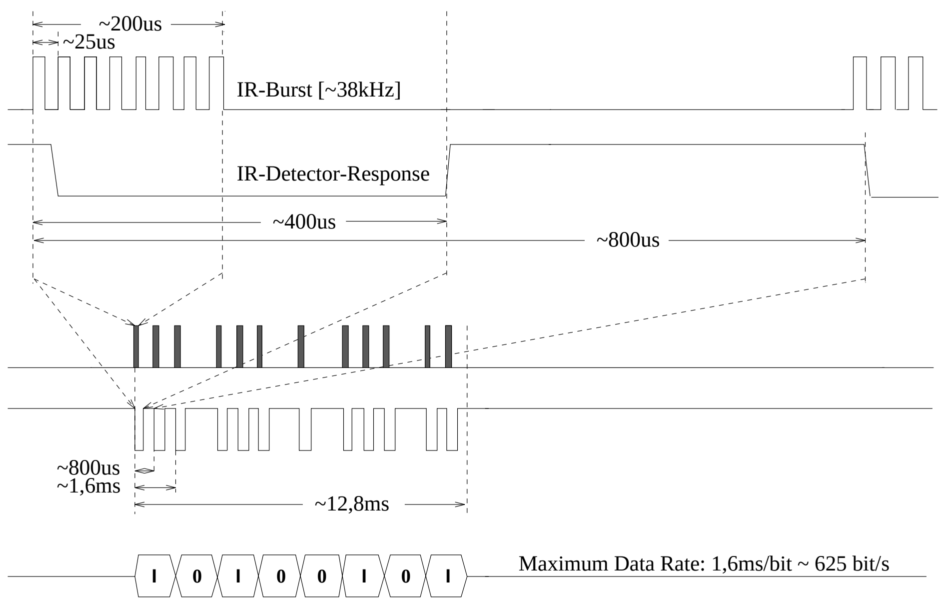

Figure 3: **Timing of Long Range Infrared Link** *On side of the infrared transmitter a finite state machine inside the on board tram controller generates the puls packet scheme which can be received by the Sharp infrared detector. The receiver FPGA reconstructs the original bit sequence from the response of the infrared detector.*

Until now, the software development (PC programming) has not met the given requirements. Though a collision free moving of different trams is possible, the applied methods can still be improved. One subgoal is the computer representation and visualization of the railway net together with the moving trams. Another important goal in this context is to allow an automatic generation of a computer model for a free configurable railway net topology. This could be achieved by an initialization run with a special tram, which delivers the necessary data for the generation of the computer model for the whole rail net.

The main problem in development of the control software was the low data transfer rate from the Zimo transmitter to the Zimo receivers on the trams (see right part of the main control loop, figure 1). The rate of the RS-232 link from the host PC to the Zimo sender is 2400 *baud*. But the Zimo system allows only 10 commands per second with a command length of 2-3 bytes. The infrared link from the trams to the PC is 600 *baud*. This results in a reaction time in the range of 500 ms. The Zimo system is the bottleneck in this control loop. But with these reaction times, the "go one after another" approach with distances in the range of millimeters can only be attained by complex steering algorithms. Those algorithms have to project the current tram movements into the near future to react with foresight. Replacement of the Zimo system by a faster communication channel – for example with a high speed duplex infrared link – would allow to simplify the algorithms.

The project described is very popular with the students and will be continued in the fall semester 94/95. However, it seems that especially those students (of computer science) participate in the project, who are interested mainly in hardware. It will be our task to convince them that one has to develop hardware and software together, in the way of codesign.

3 Conclusion

Our course points out that in addition to a fundamental theoretical education a practical one is important, to refine the theoretical knowledge by practical relevant aspects. Our choice of a

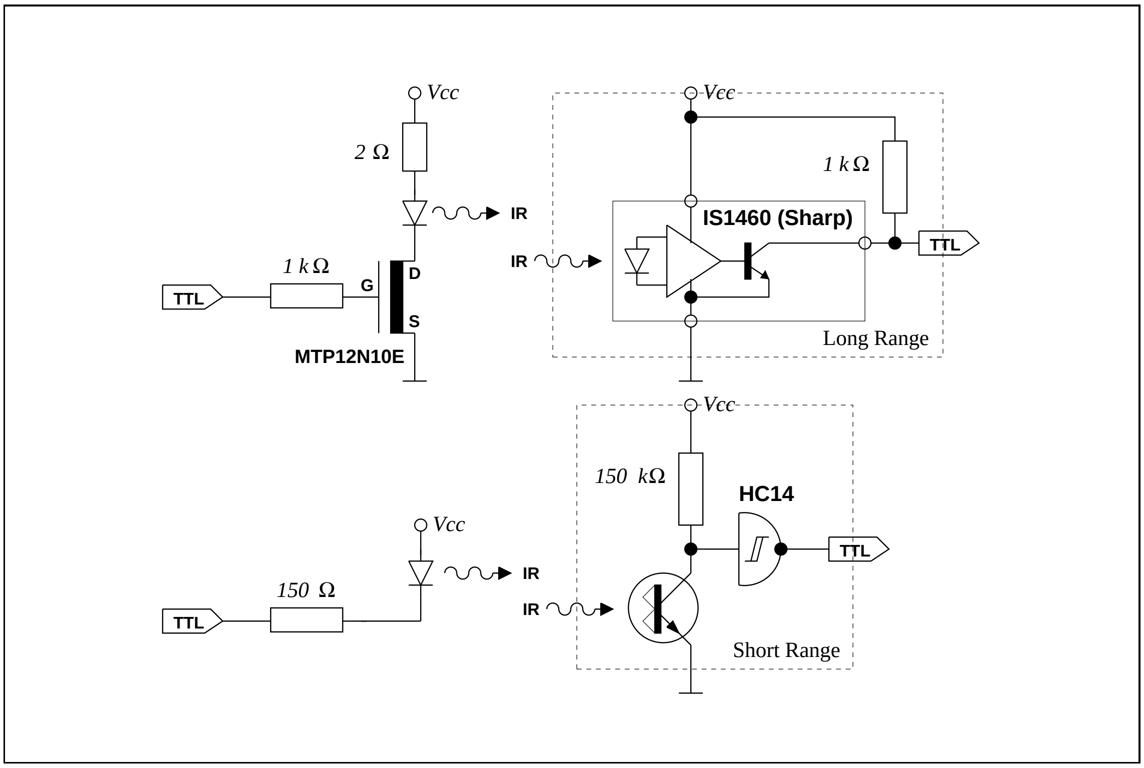

Figure 4: **Interfacing Infrared Signals**

rather simple technical process allowed to concentrate on the design flow itself with main emphasis to the integration on system level. Normally the advantage of software design is its interactive nature in development, while hardware (VLSI) design traditionally has very long design turn around times. With the availability of FPGAs and the resulting short turn around times this interactive aspect is possible for hardware design and system integration too.

Due to the strong interest of the students in this project some further activities have been started. One student designs a real time image processing system based on a neural network to steer by sight. As a support to this, another student developes a FPGA based real time video digitizer. One more student realizes a high speed infrared interface based on sophisticated operational amplifier circuits which will allow a transfer rate of $\approx 40\,kbaud$.

References

[1] A. Klindworth, *"A Tool-Set for Simulating Altera-PLDs using VHDL"*, Intern. Workshop on Field Programmable Logic, Prague 1994

[2] *Altera Databook*, Altera Corporation, San Jose CA, 1993.

[3] *IEEE Standard VHDL Language Reference Manual, VHDL Language Reference Manual*, IEEE Standard 1076-1987, New York, 1988.

[4] *Altera MAX+plusII reference manual*, Altera Corporation, San Jose CA, 1992.

[5] *Synopsys VHDL Simulator Reference Manual*, Synopsys Inc., 1993.