Abstract—This paper describes the implementation of an FPGA prototype of an application based on embedded ASIC technology. The overall goal is to implement a system that can monitor an Ethernet data stream and extracts configuration data marked by the EtherType field in the Ethernet header. For evaluation the application is implemented on a prototype consisting of two XILINX FPGA boards.

Since the target platform is an ASIC with embedded reconfigurable architectures the prototype is divided in the corresponding parts. One board emulates the embedded reconfigurable architecture that contains the Ethernet MAC. Ethernet packets can reconfigure this MAC. The second board emulates the static part of the application that controls the reconfiguration process.

I. INTRODUCTION

In high-end telecommunication network technologies a rapid evolution occurs. On the one hand, newer network technologies and standards often provide better bandwidth utilization and a higher quality of service. Network providers are interested in an early adaptation of new technologies and standards for optimal network exploitation. This requires vendors of telecommunication equipment to develop rapidly and adopt these technologies early. On the other hand, long lasting standardization processes force manufacturers into risky development strategies, since an early time to market goes along with probably unstable network standards. This causes high business risks. Design re-spins and updates of the network devices resulting from this situation can cause enormous costs.

Since the overall situation cannot be changed, a solution is required which lowers the development risks. The high data rates in current networks don’t allow the implementation in software which would be easy to update. So the reconfiguration of hardware is needed to solve the evolution problem. A theoretical approach to face with these difficulties has been presented in [1].

In this paper, we present the implementation of a prototype of an Ethernet node, which can be dynamically reconfigured using the Ethernet protocol. The Ethernet node is a prototype of a later application of a Configurable System-on-Chip (CSoC) [2]. The application shows the capabilities of its embedded reconfigurable technologies.

II. RELATED WORK

In the last decade a lot of work has been done in the field of reconfigurable computing. One aspect is to increase the chip area virtually by loading only active modules in reconfigurable area. Coarse-grained coprocessors on a scale of IP cores allow performing computations with hardware performance by retaining the flexibility of a software solution. A variety of application fields are covered such as cryptography [3], multimedia [4], [5] or even automotive industry [6] and prove this concept.

Nevertheless traditional reconfiguration overhead [7], like power consumption and reconfiguration time, needs to be taken into account as it prevents most applications from deployment in the field. The Riverside University of California has published a paper that discusses the potential reduction of energy consumption while using reconfigurable logic under the aspect of scalable supply voltage [8].

Applying runtime reconfiguration on network infrastructure has already been suggested in [9]. In [10] attention has been paid to issues on the dynamical replacement of network interface hardware while maintaining connectivity and preventing packet loss. However, the aspect of using runtime reconfiguration to adapt network devices to future protocol standards is barely addressed so far.

III. DISTRIBUTING CONFIGURATION DATA IN THE NETWORK

Integrating reconfigurable technologies in a network node allows adjustments in the hardware implementation at lifetime to patch the node or to upgrade it to a new protocol version. In that way an update of the node hardware, which is operating at line speed, becomes necessary as reconfiguring the nodes by service personal in a manual way consumes too much time and travel costs. The configuration data is transported in band (within the normal network stream) by exploring data paths in the network, so no additional interfaces on the network nodes are necessary. As shown in fig. 1 configuration data
is distributed by a centralized gateway using the broadcast mechanisms of the network.

Deployment of this method in large-scale networks requires that functionality and security issues have to be solved. On the one hand, when reconfiguring nodes in band over an unreliable network, e.g. adaptation to a new protocol standard, it has to be ensured that no configuration data is lost. After the update process the network must obtain a consistent state. If the reconfiguration fails for some reason on any node it must be ensured that it still can be accessed. On the other hand, the reconfiguration mechanism has to be protected by authorization and encryption methods against compromisation. In [11] a versatile framework for FPGA field updates is proposed that covers some important issues.

In this project Ethernet (IEEE 802.3) has been chosen as the network layer for the demonstrator implementation. As Ethernet is widely spread and uses standard devices it enables us to create a demonstrator of reasonable size. Fig. 2 shows the Ethernet frame for the in band data transport. For the demonstrator an unregistered type number in the type field of the frame is used to identify the configuration data packets. Furthermore an address field allows the reconfiguration of multiple dynamic areas in a node.

The prototype platform emulates an embedded FPGA placed on an ASIC as it is planned in a further design step. So the platform consisting of two Virtex-II Pro FPGA boards has been divided into corresponding parts. The first board emulates the embedded FPGA macro and contains an Ethernet MAC, whereas the second board represents the static part of the system that controls the reconfiguration process. Fig. 3 shows the overall emulation system.

The demonstrated application is a node connected to an Ethernet network. This node monitors the data stream received via Ethernet and identifies reconfiguration data sent by a PC on the basis of the EtherType field. The reconfiguration data is then extracted from the data stream by the Ethernet MAC on the first board and transferred via RocketIO interface to the memory of the second board. After the complete data stream has been received and the checksum verification passed to determine transmission errors the reconfiguration process is initiated.

The second board generates the reconfiguration data stream from its memory. The reconfiguration of the first board is done using the JTAG interface (IEEE 1149.1), so the memory data needs to be transferred into JTAG commands as used by the FPGA programming software on PC. One of the main difficulties for this prototype occured in implementing this JTAG reconfiguration sequence. The implementation details are shown in the next section.

The MAC can be monitored and controlled by the PowerPC core, which is included in the FPGA. The software running on this core provides access to all registers of the MAC. A separate RS232 connection between the external PC and the board allows an Ethernet independent communication so the functionality of the MAC can be checked even if the Ethernet connection has a malfunction.

To show the functionality of the reconfiguration process an initially faulty Ethernet MAC on the first board is inserted. The Ethernet packets contain a CRC32 checksum to verify their payload. Initially, the CRC check in the MAC is corrupted so verification produces wrong results. The received Ethernet packets are looped back to its source, the stimulating PC. There the error in the MAC is visualized and the service personal can initiate the bug fix process. The new Ethernet MAC with corrected CRC calculation is sent by the in band
reconfiguration process to the network node. This node loads the new hardware and the success of the reconfiguration will be visible directly on the PC when it receives valid Ethernet packets.

V. DEMONSTRATOR IMPLEMENTATION

For the implementation of the prototype the XILINX EDK 8.2i design software is used. The EDK has also been used for software development by utilizing the provided libraries for drivers and operating system.

A. Implementation of the dynamic part

The first board emulates the dynamic part of the application. It contains the Ethernet MAC and components needed to control the MAC to handle the received configuration packets. The system is divided into three main components. These are the wrapper containing the MAC and additional logic, the PowerPC core and an RS232 interface. A “Processor Local Bus” (PLB) connects these components. To make customized IP cores compatible to the PLB they have to be attached to a PLB interface (PLB-IPIF). This IP core is provided by XILINX and features parameterizable FIFO for data input and output in various bit widths.

The most important component is the Ethernet MAC wrapper, which is shown in fig. 4. Beside the Ethernet MAC it contains additional logic that handles the configuration data separation and control functionality. Furthermore this logic enables the debug functionality via PowerPC and RS232 interface.

B. Implementation of the static part

The second board emulates the static part of the application. This part consists of a PowerPC core, a DDR RAM with a size of 256 MB including a JTAG controller and a user IP to receive and to send the configuration data. In the same way as on the first board the components are connected with a PLB. The structure of the reconfiguration controller is shown in fig. 5.

After configuration data has been received, it is stored in a FIFO. This is necessary because the data is passed to the DDR RAM using burst transmission mode. After the configuration stream has been stored in the DDR RAM completely, the PowerPC starts the calculation of the CRC. If no errors are detected the PowerPC writes the register START_RECONF in the macro to initialize the reconfiguration of the first board. The macro reads the data from the memory and sends it to the first board using the JTAG interface.

The JTAG interface is split in two parts - a software part running on the PowerPC core and a hardware part generating and receiving the four JTAG signals (TDI, TDO, TMS, TCK) connected to the PLB. For the connection to the dynamic board a special cable with 4 ground wires for shielding is used to prevent transmission errors which occurred with an unshielded cable. The FPGA pins are driven using a high driver strength of 12 mA. The JTAG controller generates an own transmission clock, serializes the data provided by the software via PLB and ensures the timing at the interface as this exposes to be the most critical part of the prototype.

The software part generates JTAG instructions and encapsulates the reconfiguration data as described in [12]. Before transmitting the data stream to the first board it is necessary to check the ID- and USER-Code field and the state register to ensure configuring the right device. The data is read from DDR-RAM using a pointer incrementation procedure. After transmission of all data the state register is read again to verify the reconfiguration process.

VI. RESULTS

The hardware designs were implemented in VHDL and synthesized using Xilinx XST as part of EDK 8.2i. The implementation of board 1 requires 3700 Slices (27 % usage) and at board 2 2750 Slices (about 20 %) are used for the design.

After the reconfiguration data stream has been transmitted the board 2 reconfigured board 1. The reconfigured ethernet MAC was detected by the PC at about 2.5 seconds after sending the reconfiguration data at 750 Hz. The exact time depends on the used JTAG-TCK frequency which can be varied between 750 Hz and 25 MHz.
Two main overheads slow down the reconfiguration process. The first one is the overhead generated by the Ethernet packets as each reconfiguration packet is included into an Ethernet frame. In the prototype OSI level 4 is chosen (using UDP/IP) to use the mechanisms of the lower protocol layers. The used Ethernet MAC is reduced in its size so it only can handle the minimum packet size of 64 byte. Due to the overhead of the lower OSI layers only 20 byte of reconfiguration data can be transferred in one packet. This means that in the ideal case only 30% of the Ethernet bandwidth is used for the reconfiguration data transfer.

The other bottleneck is the speed of the JTAG transmission. Depending on the used JTAG clock frequency (TCK) and the size of the device to be reconfigured the standard reconfiguration stream of about 1.7 MByte needs at least 0.5 s transfer time.

VII. FUTURE WORK

The architectures of the emulation platform and the planned ASIC platform are quite different. The new platform will be an ASIC consisting of static and reconfigurable parts. This combines the advantages of static hardware (better power consumption and wire lengths) with the ability of flexible adaptions to new standards. Fig. 6 shows a simplified view of parts of the future chip with embedded reconfigurable areas. There will be three various reconfigurable architectures (XPP, PicoGA and FlexOS) provided by several companies on the new platform. The presented network application uses the M2000 FlexOS [13] embedded FPGA macro which will host the Ethernet MAC. The controlling microprocessor will be an ARM9 and the configuration data will be stored into system memory using the connected AMBA bus. After the configuration data has been collected completely and verified, the M2000 configuration manager takes care of the reconfiguration of the M2000 macro. For the transport of the configuration data from system memory to the M2000 macro a dedicated network on chip will be used.

In contrast to the new platform the reconfiguration procedure of the prototype differs a lot (see fig. 3). On the prototype the reconfiguration data is transmitted via RocketIO to the second board where it is stored into local memory. The reconfiguration of the FPGA on the first board is controlled by the PowerPC of the second board and transferred using the JTAG protocol. So one of the bottlenecks (JTAG transmission time) named in the results section will be eliminated in the new design.

VIII. SUMMARY

In this paper we described the reconfiguration of network nodes via Ethernet and the realization of an FPGA prototype for the node to emulate an ASIC containing a embedded reconfigurable arrays. The prototype consists of two XUP boards, where one board emulates the reconfigurable part and the other board emulates the static part of the application. The application implemented on this prototype is an Ethernet MAC, which can be reconfigured by Ethernet packets broadcast in-band by a service host.

REFERENCES