SoNIC over 1G

Adithya Venkatesh, Nandini Nagaraj, Rafael Farias Marinheiro and Han Wang
Cornell University

***Abstract*—In standard environments, both Data Link and the Physical (PHY) layers are defined in the Network Interface Cards (NIC) and they cannot be accessed in real-time via software. However, these lower layers contain valuable information that can be used to measure and to improve the performance of the network. Recently, SoNIC [1] was proposed to provide real-time access to the Physical Layer and it was used to accurately measure the performance of a high-speed wired complex network [2]. Current SoNIC design and implementation only operates in 10 GbE (Gigabit Ethernet) devices. In this project, we present a SoNIC design for 1GbE devices as an effort to extend the SoNIC ecosystem.**

I. Introduction

In standard environments, both Data Link and the Physical (PHY) layers are defined in the Network Interface Cards (NIC) and they cannot be accessed in real-time via software. However, these lower layers contain valuable information that can be used to measure and to improve the performance of the network. Recently, SoNIC [1] was proposed to provide real-time access to the Physical Layer and it was used to accurately measure the performance of a high-speed wired complex network [2]. However, current SoNIC design and implementation only operates in 10 GbE (Gigabit Ethernet) devices. In this project, we present a SoNIC design for 1GbE devices as an effort to extend the SoNIC ecosystem.

II. Background

According to the IEEE 802.3 standard [3], the PHY of the 1000BASE-X standard consists of three sublayers: the Physical Coding Sublayer (PCS), the Physical Medium Attachment (PMA) sublayer, and the Physical Medium Dependent (PMD) sublayer (see figure 1). The PMD sublayer is responsible for transmitting and receiving symbols from the medium. The PMA sublayer is responsible for clock recovery and for serializing and deserializing the bitstream. The PCS sublayer comprises the encoding and decoding scheme used in the 1000BASE-X standard. In order to support different Gigabit technologies, the PCS communicates with the Gigabit Media Independent Interface (GMII). The IEEE 802.3 Clause 36 explains the PCS sublayer in further detail, but we summarize it below.

When Ethernet frames are passed from the data link layer to the PHY, they are reformatted before being sent accross the physical medium. On the transmit (TX) path, the PCS encodes every octet of an Ethernet frame into a 10-bit *code group* using the *8B/10B transmission code* (specified in clause 36.2.4). In order to achieve DC balance[1], the 8B/10B codec uses two different code groups (RD- and RD+) for the same octet and uses a state machine to define which code group should be used. The entire 10-bit code-group is transmitted over a physical medium. On the receive (RX) path, the PCS decodes the 10-bit code-group into the octet and then sends it to the layer above.

[1]A Direct Current (DC) balanced signal has a similar number of 0's and 1's. It is used to prevent bit errors in circuits.

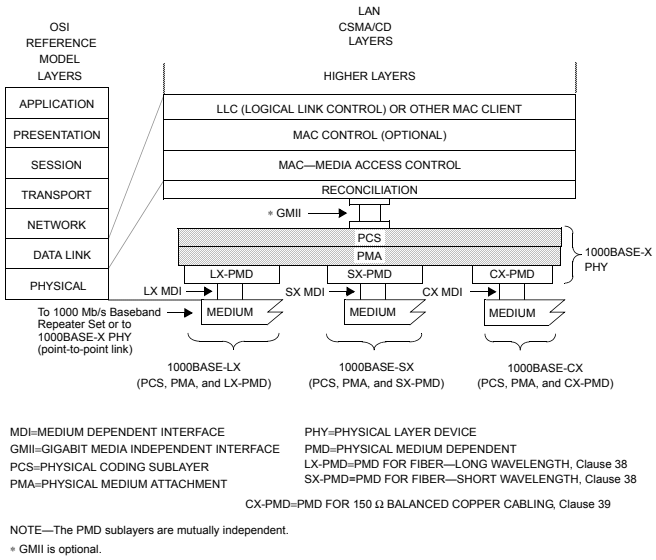

Fig. 1. 1000-BASEX standard

III. Design

In order to provide real-time access to the PHY layer in software, the user must have direct access to the PCS layer. In our solution, the functionalities of the PCS layer must be implemented in software while the transmission and reception of bits can be handled by the hardware. In figure 2, we present our design.

In this design, the PMD and the PMA sublayers are implemented in hardware. The PMA layer stores data in and receives data from the ring buffers. A Direct Memory Access (DMA) engine must also be used to transfer the data from the hardware layer to the kernel implementation. The PCS and the upper layers are implemented in software. The PCS is implemented in kernel space to access the shared memory region between the hardware and software.

In order to achieve the full line-rate of 1Gbps, the software implementation must be able to process and transfer data at 1.25Gbps, due to the 8B/10B encoding. While this does present some challenges, it is worth pointing out that this rate is much less than the 10.3125Gbps required by the 10GbE standard.

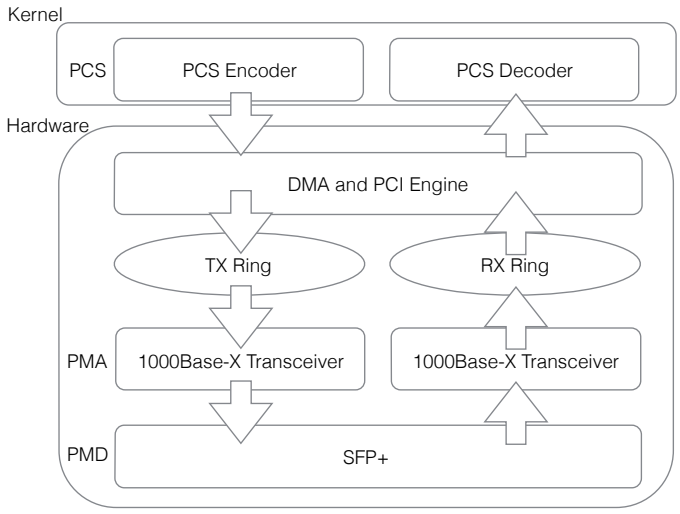

Fig. 2. SoNIC over 1G Architecture

IV. Implementation

A. Hardware

To implement the hardware part of the design, we planned to use the Altera Stratix IV FPGA as our platform. The board is already equipped with SFP+ (Small Form-factor Pluggable) ports and it is equipped with 11.3 Gbps transceivers that can easily perform the 1 GbE PMA at line-speed.

We have implemented the PMA sublayer and the Buffer Rings in Verilog. We didn't manage to implement the DMA engine in hardware, but we can easily reuse the original SoNIC DMA implementation to transfer data between the Hardware Layer and the Software Layer.

B. 8B/10B Transmission Code

1G SoNIC requires an 8B/10B encoder/decoder to be available to the Physical Coding Sublayer (PCS). The 8B/10B encoder converts an octet input into a 10-bit code group, the decoder does the opposite. Valid inputs to the codec are available in the [3], and each octet input maps to one of two 10-bit outputs depending on the running disparity of the codec (difference between the number of set and unset bits in each input to the encoder). The 8B/10B Codec for 1G SoNIC was implemented in C. The software implementation works as follows. It has two 3/4 lookup tables and two 5/6 lookup tables, one for the positive running disparity and the other for the negative running disparity, containing 8 and 32 entries respectively. The decoder works with a reverse lookup table and does not have to keep track of the running disparity.

Decoding happens between the PCS and the Gigabit Media Independent Interface (GMII). Encoding happens between the PCS and the Physical Medium Attachment (PMA). The codec is required to allow the clock to recover between states. This relaxation is provided by adding the 2 extra bits which increase the time required to process by 25%.

While the implementation was straight-forward it was necessary to ensure that the codec adhered to the 1Gbps line-rate of the device. The initial implementation used the right

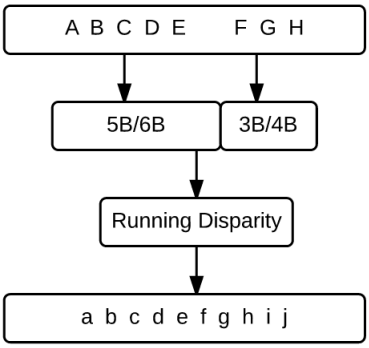

Fig. 3. Encoder

shift operator in C to count the number of bits set to 1 in the 8-bit input, however this proved to be too computationally expensive to let the codec meet the 1Gbps line-rate. The builtin popcount method, provided by GCC combined with compiler optimizations, proved to be the key for efficient computation of the number of bits in the input byte and let the program meet the line-rate requirements.

V. Evaluation and Validation

A. Hardware Implementation

We have tested our hardware implementation against a testbench available in [4]. Our implementation passed all tests provided.

B. 8B/10B Transmission Code

1) Performance: The 8B/10B codec was evaluated to measure the rate at which it processes data and to ensure that it met the 1Gbps requirement. The encoder takes an 8-bit input, 1Gb is $1024^3 = 1073741824$ bits, or 134217728 bytes. An encoder that meets the line-rate requirement would process this amount of data withing a 1 second window. In the same vein, a decoder that performs at line-rate would process 107374183 10-bit inputs within this 1 second. The encoder and decoder were both tested for varying input sizes in increments of 1Gb for each test and the results are available in figure 4 and in figure 5.

2) Correctness: A correct codec would process data as specified in the Table 36 of the IEEE Standard for Ethernet [3], this could be tested with a simple script. Moreover when run in this sequence $y \leftarrow encode(x)$ followed by $z \leftarrow decode(y)$, a correct encoder would produce z such that $x = z$, if $x \neq z$ - the codec is performing incorrectly. The codec passed both tests with the value of running disparity being hard coded to $+1 \& -1$. The codec also exposes a function $self\_test\_8b10b$ that accepts two parameters, an array of 8-bit values and the length of the array, that fails if the encoder fails to perform correctly.

VI. Conclusion

In this project, we have designed a 1GbE version of SoNIC. We have developed a working implementation of the encoding/decoding of the physical layer of 1GbE in the software and also a partial working implementation of the hardware layer.

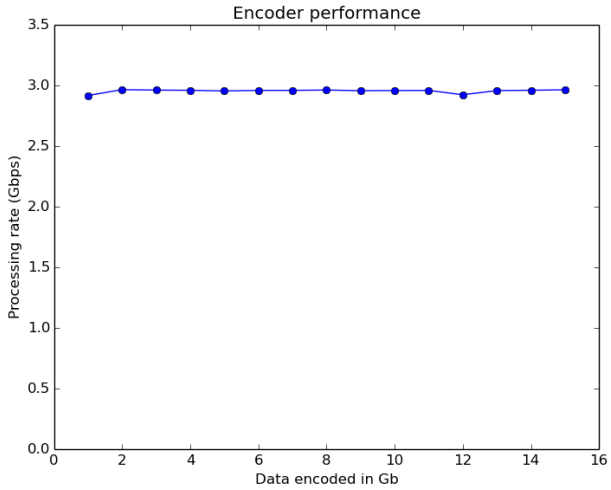

Fig. 4. Encoder Performance

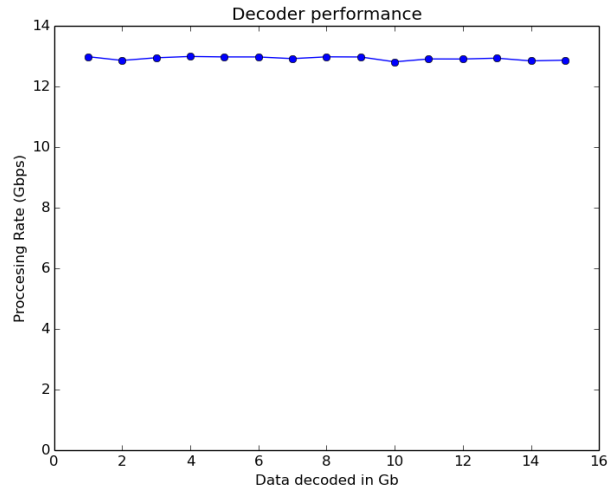

Fig. 5. Decoder Performance

We hope that this effort contributes to the extension the SoNIC ecosystem, allowing more system programmers to use it as a valuable tool for cross-network-layer research.

REFERENCES

[1] K. S. Lee, H. Wang, and H. Weatherspoon, "Sonic: Precise realtime software access and control of wired networks." in *NSDI*, 2013, pp. 213–225.

[2] H. Wang, K. S. Lee, E. Li, C. L. Lim, A. Tang, and H. Weatherspoon, "Timing is everything: Accurate, minimum overhead, available bandwidth estimation in high-speed wired networks," in *Proceedings of the 2014 Conference on Internet Measurement Conference*. ACM, 2014, pp. 407–420.

[3] "Ieee standard for ethernet - section 3," *IEEE Std 802.3-2012 (Revision to IEEE Std 802.3-2008)*, pp. 1–0, Dec 2012.

[4] "1000base-x ieee 802.3-2008 clause 36/37 - pcs testbench," http://opencores.org/websvn,filedetails?repname=1000base-x&path=%2F1000base-x%2Ftrunk%2Ftestbench%2FREADME&rev=6, accessed: 2014-12-12.