Design and Implementation of a SNTP Client on FPGA

J. Viejo, J. Juan, M. J. Bellido, E. Ostua, A. Millan, P. Ruiz-de-Clavijo, A. Muñoz, and D. Guerrero
Grupo ID2 (Investigacion y Desarrollo Digital)
Departamento de Tecnologia Electronica-Universidad de Sevilla
E. T. S. Ing. Informatica, Campus Universitario Reina Mercedes
41012 Sevilla (SPAIN)
Email: julian@dte.us.es, jjchico@dte.us.es, bellido@dte.us.es, ostua@dte.us.es,
amillan@dte.us.es, paulino@dte.us.es, amrivera@dte.us.es, guerre@dte.us.es

***Abstract*— This contribution presents the design and implementation of a SNTP client module suitable for IEC 61850 environments fully done in hardware. The module is able to provide synchronization and accurate time reference within a microsecond with respect to a SNTP server, in a extremely compact, cost-effective and low power device completely implemented in a low grade FPGA chip. Therefore it can be an ideal replacement to expensive computer-based solutions or dedicated GPS receivers in a wide range of industrial applications. This SNTP client is part of a common technological platform for implementing Remote Terminal Units (RTUs) under IEC 61850.**

I. Introduction

Time stamping is a critical task in many industrial control systems. Data acquisition by Remote Terminal Units (RTUs) being a typical example. In this sense, the industry norm IEC 61850 [1] defines the Simple Network Time Protocol (SNTP) [2] over Ethernet as a standard way to synchronize a set of substations with a time server. SNTP is a simplified version of the more general Network Time Protocol (NTP) [3] that is commonly used in Internet servers and routers. Both SNTP and NTP share the same communication protocol and data format, the main difference being that NTP uses sophisticated algorithms that ensures a correct synchronization with multiple servers under highly variable latency data links, which is common in a world wide network like the Internet. On the contrary, SNTP covers the synchronization with a single server and uses a simplified stateless algorithm, which makes it suitable for embedded systems in a controlled industrial environment. Nevertheless, a SNTP client may communicate with either a SNTP server or a full NTP server.

In this scenario, a time server will typically gather accurate time information from an absolute reference like an accurate clock or a GPS receiver. SNTP clients located at the substations will synchronize with the server through the Local Area Network (LAN) making it unnecessary to install absolute references at the substations. SNTP clients will then provide the nearby electronic equipment with the necessary time information.

This contribution is part of a project supported by the Ministry of Industry of Spain and leaded by the Telvent company [4] which finality is to develop a Common Technological Platform (PTC) to facilitate the implementation of the functionality typically found in Remote Terminal Units (RTU) used to control the public power grid. A key point in the project is to assure the synchronization of electronic equipment within the range of a few microseconds. This synchronization is achieved by the use of SNTP clients and servers fully implemented in hardware. SNTP client and server design is divided in three main phases:

1) Basic protocol stack implementation and integration with Ethernet controller. At least the following protocols are needed: IP, ARP, UDP, and NTP.
2) NTP client implementation: issuing of requests, answer processing and local clock synchronization.
3) NTP server implementation: synchronization with an external reference and request processing.

This paper describes the current status of the client implementation that includes phase 1 and part of phase 2.

The rest of this contribution is organized as follows: a brief introduction to NTP and SNTP is included in section II, section III enumerates the requirements and general specifications of the system, section IV gives the details of the design and implementation of the hardware SNTP client, section V includes some implementation results and section VI discusses some conclusions.

II. NTP/SNTP protocol basic operation

The operation of the NTP/SNTP protocol is very simple (Fig. 1). The client sends a request to the server by issuing an UDP data packet where the time of its local clock (T1) is included. When the request is received at the server a new time stamp T2 is generated with the reception time as given by the server's local clock. After processing the request, the server issues a reply including the time at which the reply leaves the server (T3). When the client receives the reply, the arrival time (T4) is also annotated. With this set of timestamps the client can calculate the round trip time (t_{rd}) and the time offset between the client's and server's clocks (t_{offset}). Assuming a symmetric connection it gives:

$$\begin{aligned} t_{rd} &= (T_4 - T_1) - (T_3 - T_2) \\ t_{offset} &= \frac{(T_2 - T_1) + (T_3 - T_4)}{2} \end{aligned} \quad (1)$$

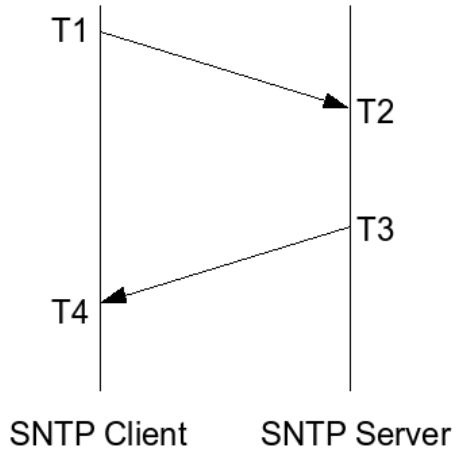

Fig. 1. Operation of the NTP/SNTP protocol.

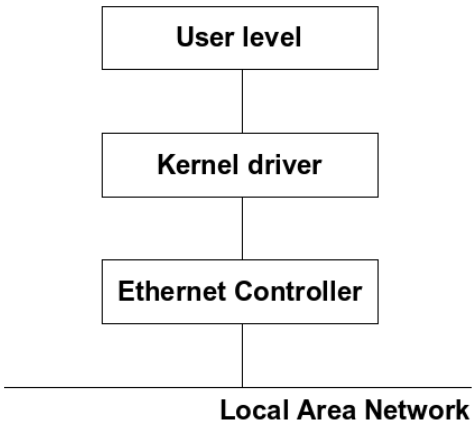

Fig. 2. Layers where NTP can be implemented.

Using the calculated offset, the client can correct its local clock to match the server time. Software implementations of NTP tipically achieve time synchronization within a millisecond with respect to the server [3]. There are two main sources of error. The first one is the asymmetry in the network communication when the time spent by the client's request to reach the server is different to the time spent by the answer to reach the client. This is due to unpredictable latency in network equipment, specially when collisions take place and the number of the devices involved increases. The second main source of error is due to the variable time gap between the instant the time stamp is registered in the datagram and the real instant the datagram leaves or reaches the host. In typical software implementation, these time stamps are registered by client/server software running as a user level application (Fig. 2) so the time stamp error will depend on the time spent processing the datagram as it goes through the protocol stack and software layers. This error will largely depend on system load, detailed software implementation, etc. The precision of the NTP synchronization can be largely improved by doing the time-stamping operation in lower layers [5], therefore some operating system kernels like Linux of FreeBSD support NTP processing in the kernel [6]. This way, precision may reach some tens of microseconds.

The highest precision in the time-stamping operation is only achievable if done by the Ethernet device hardware as soon as the packets arrive or leave the interface.

III. System Specification

The main objective of this contribution is to build cost-effective, autonomous, compact and highly accurate SNTP client and server modules suitable for, but not limited to, IEC 61850 environments. The SNTP server will use a standard GPS receiver as a time reference. It will use the PPS (Pulse Per Second) signal and NMEA data from the GPS to synchronize its internal clock. The SNTP client will synchronize with the server through the local area network using the NTP protocol, and will provide a PPS signal and NMEA information through a serial interface thus emulating a GPS receiver. A typical scenario is depicted in Fig. 3 where the server (H-SNTPD) gathers the time from the GPS receiver and clients (H-SNTPC) provide time and synchronization information to remote terminal units (RTU).

More specifically, the SNTP client should meet the following criteria:

- The client will operate in a standard 10/100/1000 MHz Ethernet LAN.
- The client will be configured automatically using the BOOTP [7] protocol so that the configuration for all the clients can be centralized in a single BOOTP server.
- The precision of the local clock at the client should be within 10 μs with respect to the server's clock in optimal conditions: hardware time-stamping in the server and direct LAN connection without switches. In typical conditions (software server and standard switch connection) precision should be always within 1 ms.
- The whole client design should fit in a single, low density FPGA chip and should need no additional hardware, so that cost of system parts will be under $20 before mass production.
- Low power. Implemented in a low density, low frequency FPGA, the client will consume under 1 W of average power which is much lower than a computer-based implementation that would consume about 100 W.

IV. Design and implementation

In this section, the most important aspects of design and implementation are commented. A diagram of the modules that form the SNTP client is shown in Fig. 4. We can distinguish the following parts: control unit, Ethernet MAC controller, SNTP client module, and PPS generation and RMC frame transmission module. Next, we will briefly explain the functionality of each one of these subsystems.

The control unit is in charge of arbitrating the operation of the rest of the modules in order to perform the adequate task in each moment. The module has been modeled as a Finite State Machine using Verilog coding according to the structure

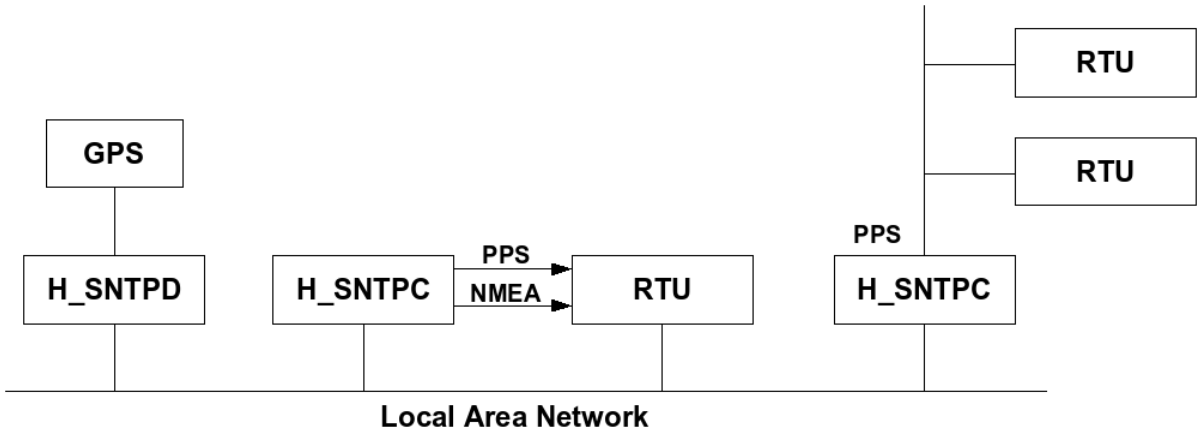

Fig. 3. Typical scenario for deploying hardware SNTP client and servers.

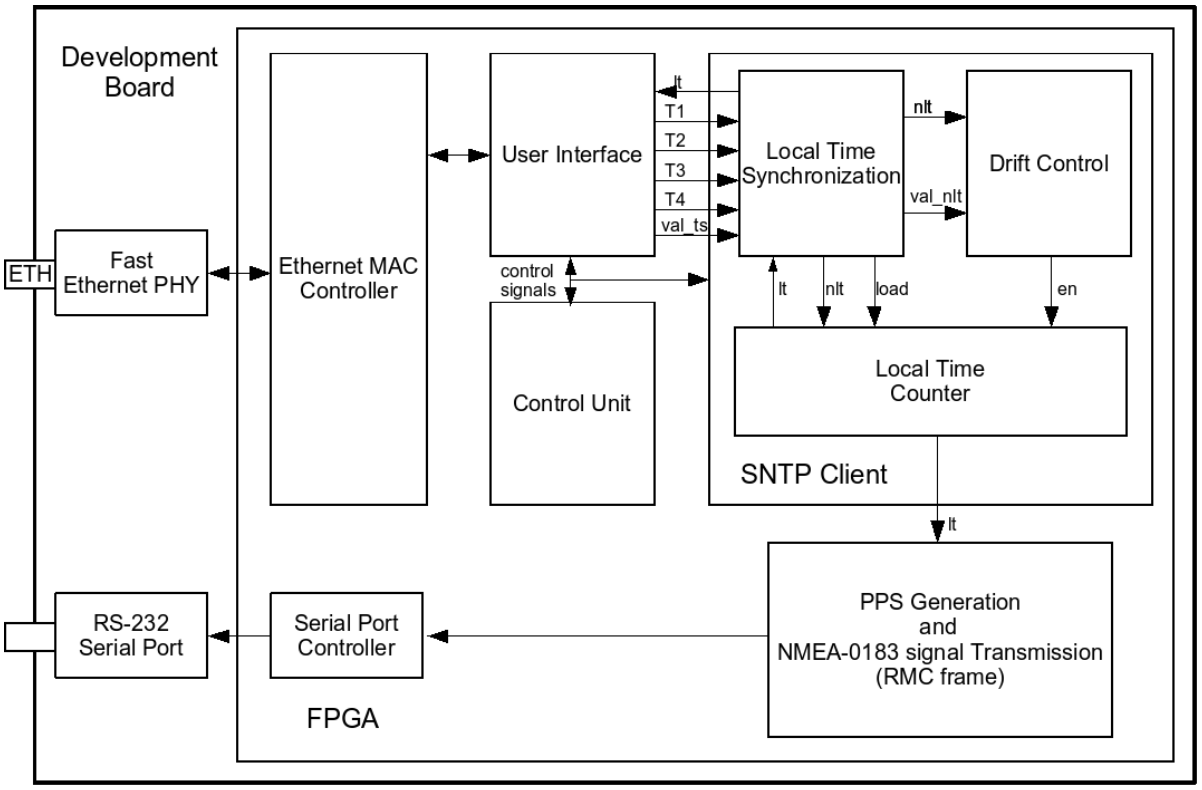

Fig. 4. Block diagram of the SNTP client.

described in [8]. The control unit defines two main operating modes:

1) Configuration. When the SNTP client starts to operate or after a system reset, an automatic configuration process is performed according to the Bootstrap Protocol (BOOTP) [7]. This process consists of finding the SNTP client MAC address and a series of configuration parameters like SNTP client and server IP addresses and the RS-232 serial port baudrate. We have used BOOTP because of its simplicity compared to DHCP [9] which makes it more suitable to be implemented in hardware, while the extended capabilities of DHCP are not useful for the intended application and would only introduce extra development and resource costs.
2) Normal operation. Once the configuration process has finished, the SNTP client begins to work into the normal operation mode. In this mode, the device carries out different tasks which we are going to summarize next. Firstly, the SNTP client needs to know the SNTP server MAC address, so the client includes a simple implementation of the Address Resolution Protocol (ARP) [10]; as well, the client must be able to send an ARP reply packet

whenever another device requests its MAC address. Secondly, the client must transmit a time request packet (SNTP message) at secondly intervals. Finally, when the SNTP client receives the time reply packet, the timestamps obtained are registered so that the SNTP client module can synchronize the local clock.

The Ethernet MAC controller is in charge of controlling a standard Fast Ethernet PHY device, allowing us to transmit and receive Ethernet frames according to IEEE 802.3 specification. The implementation of this module has been carried out using the Tri-mode Ethernet MAC IP-core available from the OpenCores project in its web portal *opencores.org*. Moreover, this IP-core has a FIFO interface to user applications which facilitates the SNTP client design.

The user interface has been developed according to the specification document [11]. This interface has been implemented as a finite state machine coded in Verilog, and is formed by a transmitter module and a receiver module. So, on the one hand, the transmitter module is able to transmit three different Ethernet frames: BOOTP Request, ARP Request/Reply and time request packets, which are stored in a RAM. This module also includes a memory updating component that is in charge of updating the different packet fields before transmitting them. On the other hand, the receiver module is able to identify the following frames: BOOTP Reply, ARP Request/Reply and time reply packets, ruling out the rest of Ethernet frames.

The SNTP client module is in charge of calculating the clock offset using the timestamps and synchronizing the local time. Additionally, a drift control is carried out in order to improve the local clock accuracy. At this moment, this component is being developed using the System Level tool System Generator for DSP according to the methodology presented in [12].

Finally, the PPS generation and RMC frame transmission module is in charge of generating a synchronization signal (PPS+NMEA) which will be sent through the serial port to a Remote Terminal Unit. To implement this module the same methodology used to build the SNTP client module will be employed.

At the time of writing, the system is fully specified and the most critical and complex parts have been implemented: the control unit, the user interface and part of the SNTP client functions.

V. Results

In this section, simulation and hardware implementation results are described in some detail.

A. Simulation results

In order to check that the designs works correctly, the following simulation process has been carried out. At the first stage, the design has been verified using Simulink and ModelSim. For the generation of the input stimuli, the Source Blockset of Simulink has been employed. At the second stage, we have used the Xilinx tool ChipScope Pro to perform the on-chip verification of the client. In this way, we have verified the correct transmission and reception of the different packet types: BOOTP, ARP, and SNTP messages.

TABLE I
Hardware implementation results on Spartan-3E XC3S500E.

Figure of merit	Usage (%)
Slices	1,122 (24%)
Slice Flip Flops	1,287 (13%)
4 input LUTs	1,303 (13%)
Bonded IOBs	35 (15%)
Block RAMs	10 (50%)
GCLKs	6 (25%)
Maximum operation frequency	96 MHz

B. Hardware implementation results

In this subsection, hardware implementation results will be presented. Specifically, two figures of merit will be analyzed: hardware resources and maximum operation frequency.

The design has been implemented on a Spartan-3E XC3S500E FPGA. The Table I shows the design results in the current development stage. It is worth no note that although the implementation is not finished, the most resource consuming blocks are implemented, which are the user interface and the control logic. Considering this and the fact that there is still plenty of room for optimization in the already implemented hardware, it is expected that the final resource requirements will be very close to the data in Table I.

VI. Conclusion

The design of a SNTP client completely done in hardware has been presented. By using a high level methodology and standard FPGA technology it is possible to produce a high accurate, cost-effective and flexible solution for accurate time distribution and time stamping in industrial environments that agrees with international standards. First prototypes are expected to provide synchronization in the range of the microsecond in a compact and cheap device that would substitute expensive computer-based solutions or dedicated GPS receivers.

Acknowledgment

This work has been partially supported by the PROFIT-MITC PTC FIT-330100-2006-60 project and the Andalusian Regional Government's EXC-2005-TIC-1023 project.

References

[1] IEC 61850 Communication Networks and Systems In Substations, Technical Committee 57. International Electrotechnical Commission.
[2] D. L. Mills, *Simple Network Time Protocol (SNTP) Version 4 for IPv4, IPv6 and OSI*, RFC 4330, Category: Informational. January 2006.
[3] D. L. Mills, *Network Time Protocol (Version 3) Specification, Implementation and Analysis*, RFC 1305, Status: Draft Standard. March 2006.
[4] *Telvent Company Web Portal*. http://www.telvent.com.
[5] T. Skeie, S. Johannessen, and Ø. Holmeide, *Highly Accurate Time Synchronization over Switched Ethernet*, 8th IEEE International Conference on Emerging Technologies and Factory Automation, ETFA 2001, Antibes-Juan les Pins, France October 2001.

[6] D. L. Mills and P. H. Kamp. *The nanokernel*, Proc. Precision Time and Time Interval (PTTI) Applications and Planning Meeting Reston VA, November 2000.

[7] B. Croft and J. Gilmore, *Bootstrap Protocol (BOOTP)*, RFC 951, September 1985.

[8] C. E. Cummings, *The Fundamentals of Efficient Synthesizable Finite State Machine Design using NC-Verilog and BuildGates*, International Cadence Usergroup conference, ICU 2002, San Jose, California, September 2002.

[9] R. Droms, *Dynamic Host Configuration Protocol*, RFC 2131, March 1997.

[10] D. C. Plummer, *An Ethernet Address Resolution Protocol*, RFC 826, a.k.a. STD 37, November 1982.

[11] J. Gao, *10_100_1000 Mbps Tri-mode Ethernet MAC Specification*, OPENCORES.ORG, January 2006.

[12] J. Viejo, M. J. Bellido, A. Millan, E. Ostua, J. Juan, P. Ruiz-de-Clavijo, and D. Guerrero, *Efficient design and implementation on FPGA of a MicroBlaze peripheral for processing direct electrical networks measurements*, First IEEE Symposium on Industrial Embedded Systems, IES 2006, Antibes-Juan les Pins, France, October 2006.