AN FPGA BASED MULTIPROCESSING CPU FOR BEAM SYNCHRONOUS TIMING IN CERN'S SPS AND LHC

D. Domínguez, J.J. Gras, J. Lewis, J.J. Savioz, J. Serrano, CERN, Geneva, Switzerland,
F.J. Ballester, UPV, Valencia, Spain

Abstract

The Beam Synchronous Timing system (BST) will be used around the LHC and its injector, the SPS, to broadcast timing messages and synchronize actions with the beam in different receivers. To achieve beam synchronization, the BST Master card encodes messages using the bunch clock, with a nominal value of 40.079 MHz for the LHC. These messages are produced by a set of tasks every revolution period, which is every 89 μs for the LHC and every 23 μs for the SPS, therefore imposing a hard real-time constraint on the system. To achieve determinism, the BST Master uses a dedicated CPU inside its main Field Programmable Gate Array (FPGA) featuring zero-delay hardware task switching and a reduced instruction set. This paper describes the BST Master card, stressing the main FPGA design, as well as the associated software, including the LynxOS driver and the tailor-made assembler.

BST SYSTEM

The BST system will be used around the SPS and LHC for beam instrumentation and beam synchronization of distant parts of the machine. This implies the transmission through a single distribution network of a message containing the machine events, the machine status, the Coordinated Universal Time (UTC), beam synchronous commands (injection warnings, acquisition triggers, real time settings, post mortem triggers) and both, the bunch clock (up to 40.079MHz in the SPS and the LHC) and the revolution frequency (close to 11kHz for the LHC and 43kHz for the SPS) with a maximum overall jitter of 1ns (rms) between different receivers (whole system jitter). The message is therefore composed of three different kinds of information:

- One that has to be sent every revolution period (such as triggers).
- One that has to be sent occasionally (such as UTC)
- One related to the General Machine Timing (GMT) telegram [4].

The previous BST system used in the Large Electron-Positron collider (LEP) was based on a set of tasks running in the VME CPU to set up the message. Every beam revolution, the CPU had to store the new message into a VME board that contained a FIFO and was responsible for sending out each message. This methodology yields too high a CPU and VME bus usage, making it impossible to share its crate with other applications running in the Central Beam and Cycle Manager (CBCM) [1]. Besides this VME bottleneck the determinism of the system was also endangered, since each VME cycle might be delayed by interrupts, etc. By implementing our own processor in VHDL and having one processor per card, we can optimize its architecture and instruction set to achieve maximum performance and determinism. The new system's distribution network is based on the Timing, Trigger and Control (TTC) electronics and optics [2], providing a beam synchronous timing with a 25ps jitter in each receiver (local jitter). There are four different BST networks, one for the SPS, one for each ring of the LHC and a fourth one for the bunch clock sent to the experiments. Each one of these networks is driven by a BST Central Timing Generator (CTG-BST) card, which is an autonomous VME board containing a simple microprocessor (BST-CPU) implemented in an FPGA, sitting in one of the CBCM crates and therefore allowing the CBCM applications to insert the needed telegram into the BST message. A UTC block which is synchronized with the Pulse Per Second (PPS) and the 40.000 MHz signals from the GMT network, updates the onboard UTC counter after the proper initialisation and setup done by the CBCM. The output message is then sent to a TTCvx/TTCex [5] board using DC coupled ECL logic at a bitrate close to 40.079 Mbit/s, where it is then encoded and distributed optically towards the TTCrx/Beam OBservation Receivers (BOBr) [3] located in many VME crates spread around the LHC and which are responsible for monitoring and recovering the bunch clock locally.

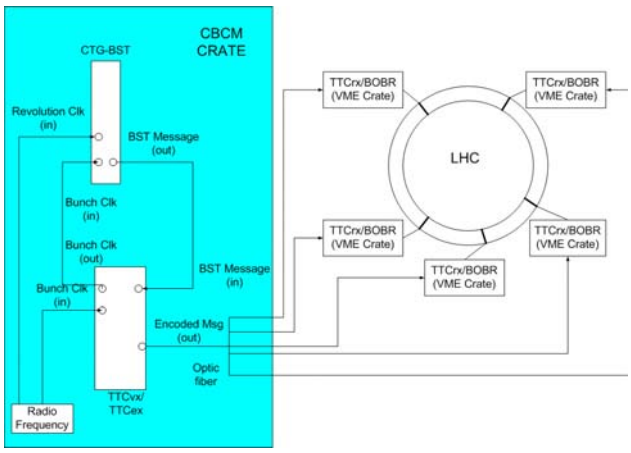

Figure 1: CBCM and BST System

BST MASTER OVERVIEW

The BST master board has two Xilinx FPGA chips onboard. The main FPGA (XC2S200 from the Spartan2 family) contains the whole functionality of the BST master, whereas the small one (XC2S50) implements a VME interface to allow downloading new versions of the

main FPGA configuration bit stream file into the main chip remotely. The new code is first sent to the board by issuing a software command in the CBCM crate and is afterwards stored in a FLASH memory on board.

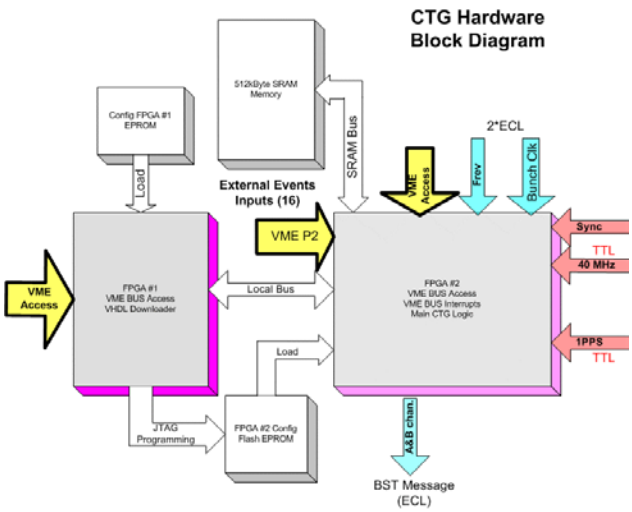

Figure 2: Overview of the CTG-BST card.

Once the CTG board is properly configured, the tasks responsible for building the message are downloaded into the 512kBytes SRAM in the CTG-BST, where the local microprocessor executes them as soon as they are enabled. A maximum of 8 tasks might be running at the same time. A round robin scheduler selects which one among them is to be processed in the next execution cycle. The implemented instruction set will allow these tasks to perform calculations on local registers, wait for external triggers, for a certain turn number or for other tasks to set an internal trigger (inter task synchronization), generate a VME interrupt or build the message to be sent during the next revolution period.

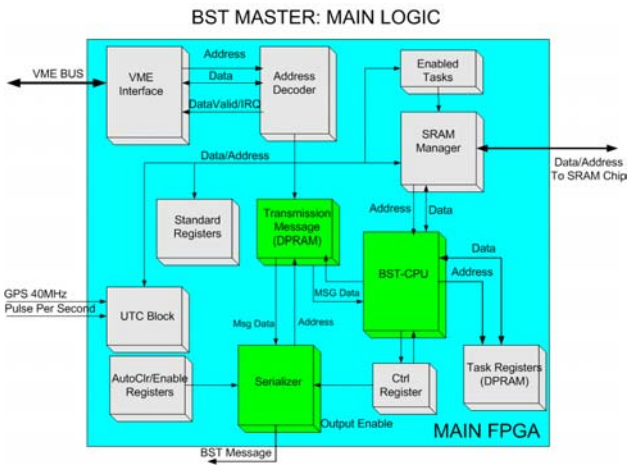

Figure 3: Architecture of the BST master Logic.

The main FPGA design is composed of:

- The BST-CPU,
- A serializer that sends the message out and implements the TTC frame format,
- A Dual Port RAM (DPRAM) to store the present and the next message,
- A second DPRAM containing the 64 task registers (8 per task),
- A UTC block,
- A VME interface and its associated registers.

All except the UTC block, run with the bunch clock.

There are two message buffers in the main FPGA: one which is being sent in the present revolution cycle and a second one being constructed for the next revolution period. The next turn message is built by the CBCM (telegram information) and by the tasks running in the local CPU. Both the crate CPU and its VME bus therefore remain available most of the time for non BST applications in the CBCM. The double buffer mechanism is completely transparent for the tasks and the user

Embedded Processor

The processor design is composed of a scheduler that uses a round robin algorithm among the enabled tasks, a control unit, a fetch unit (in charge of getting the proper instruction from the SRAM), a decoder unit responsible of decoding and fetching the operands, and an execution block that implements the operation itself and stores the result.

Each of the tasks can be enabled or disabled from the CBCM. The switching between tasks is on a single instruction basis, taking advantage of the "no overhead switching procedure". Each task can access eight 16-bit local registers (with different offsets in the DPRAM for each task) and three 16-bit global registers (turn number, external triggers and internal triggers). The turn number is used to stamp messages or for a task to wait for a certain turn number (see instruction set below). The internal triggers synchronize different tasks, whereas the external triggers are used to react to external conditions arriving at the board through the VME P2 connector. A status register informs the CBCM of each task's current situation (task running, stopped, waiting, illegal command, value or register).

The instruction set can be divided into five different categories:

- Arithmetic and logical operations (Load, Increment, Logical OR, etc.),
- Program counter instructions (Jump, branch conditions),
- Interrupt instructions (launch VME interrupt cycles),
- Message transmission operations,
- Wait instructions (wait for an external condition, an internal trigger, a relative or an absolute turn number, the arrival of the Pulse Per Second (PPS), or the arrival of a Sync pulse.

Message Transmission

The DPRAM containing the message is accessed by three different entities: the serializer (responsible of sending out the present cycle buffer), the BST-CPU (responsible for setting up most of the next revolution message), and the CBCM using VME access to provide the telegram information or to monitor either buffer. The CBCM has read and write access to both buffers whereas the BST-CPU has read access to the buffer being sent and write access to the next cycle buffer. The serializer only reads the buffer being sent.

The BST message is composed of 8-bit data words that have to be sent on a one data byte per TTC frame basis. The TTC frame format states that the destination address has to be specified for the data byte being sent in the present frame, allowing the BST master not to transmit all the bytes of a given message. As an example, UTC doesn't have to be transmitted every 23μs (SPS revolution period). An enable bit associated with each data byte selects whether to send the corresponding frame. The present system provides 32 enable bits, giving the possibility to have a 32-byte long message. When a task running in the BST CPU executes a transmission command, the appropriate enable bit is automatically set.

On the other hand, some of the BST commands/UTC information might be sent continuously or only in the next revolution message. These two modes of operation are specified by 32 "auto-clear" bits, one for each data byte.

To maximize the flexibility of the BST master, both the message length and the UTC position inside the message are programmable via VME registers. For future applications, the UTC might not be used in the message. The selected behaviour is configured via a register.

Software Environment

A complete software environment has been developed to pilot, test and program the BST master. A LynxOS driver provides a real time interface to the BST master memory map, to the interrupts it might generate and to the remote configuration of the main FPGA (downloading of new configuration bit stream file); a custom assembler, compiler and linker give the user the capability of coding the tasks to be executed in the onboard processor; a test program allows the configuration, validation and debugging of the whole hardware/software system.

PRESENT PERFORMANCE AND FUTURE IMPROVEMENTS

A basic system has been set-up to prove the feasibility and the performance of the BST master system.

A TTCex board has been used to generate the bunch clock (very stable 40.079MHz), encode the message and send it to a BOBr mezzanine card through an optical fibre.

The system has issued a maximum message length of 18 data bytes for an SPS orbit operation (revolution period: 23μs), transmitting continuously all the bytes in the message. In the case of the LHC, messages up to 32-bytes long can be sent. This represents an improvement of 10 bytes (225%) with respect to the previous BST system. The processor features 500 ns execution time per instruction, which will be reduced to 100 ns or less by pipelining and minimizing waiting states.

The utilization ratio for the main FPGA is close to 98%, a three times bigger chip will therefore be used (XC2S600E) in the next card revision. As a collateral effect, the high amount of embedded RAM in the new FPGA will allow an extra 30kByte DPRAM instantiation to store the tasks code (instead of the external SRAM chip), speeding up the reading and writing of the instructions and avoiding the need of arbitration between the BST CPU and the VME to access the memory. A more complete instruction set for the processor is foreseen in the near future.

REFERENCES

[1] Julian Lewis et al. "The Evolution of the CERN SPS Timing System for the LHC Era", ICALEPCS 2003, Gyeongju, Korea, October 2003.

[2] Joao Varela "Timing and Synchronization in the LHC Experiments", 6th Workshop on Electronics for LHC Experiments, Krakow, 11-15 September 2000

[3] Jean-Jacques Savioz "Engineering Specifications BOBr The Beam Synchronous Timing Receiver Interface For The Beam Observations", http://www.cern.ch/TTC/BOBRspec.pdf

[4] Julian Lewis et al. "The Central Beam and Cycle Management of the CERN Accelerator Complex", ICALEPCS-99, Trieste, Italy.

[5] http://ttc.web.cern.ch/ttc/intro.html