THE TIMING MASTER FOR THE FAIR ACCELERATOR FACILITY

R.C. Bär, T. Fleck, M. Kreider, S. Mauro
GSI Helmholtz Centre for Heavy Ion Research, Darmstadt, Germany

Abstract

One central design feature of the FAIR accelerator complex is a high level of parallel beam operation, imposing ambitious demands on the timing and management of accelerator cycles. Several linear accelerators, synchrotrons, storage rings and beam lines have to be controlled and reconfigured for each beam production chain on a pulse-to-pulse basis, with cycle lengths ranging from 20 ms to several hours. This implies initialization, synchronization of equipment on the time scale down to the ns level, interdependencies, multiple paths and contingency actions like emergency beam dump scenarios.

The FAIR timing system will be based on White Rabbit [1] network technology, implementing a central Timing Master (TM) unit to orchestrate all machines. The TM is subdivided into separate functional blocks: the Clock Master, which deals with time and clock sources and their distribution over WR, the Management Master, which administrates all WR timing receivers, and the Data Master, which schedules and coordinates machine instructions and broadcasts them over the WR network.

The TM triggers equipment actions based on the transmitted execution time. Since latencies in the low µs range are required, this paper investigates the possibilities of parallelisation in programmable hardware and discusses the benefits to either a distributed or monolithic timing master architecture. The proposed FPGA based TM will meet said timing requirements while providing fast reaction to interlocks and internal events and offers parallel processing of multiple signals and state machines.

FUNCTIONAL BLOCKS

Clock Master

The Clock Master is the topmost White Rabbit timing node in the system. It is the time and clock reference for all connected nodes [1] and is itself connected to a UTC source (GPS) and FAIRs RF clock system, BuTiS. It propagates absolute time and its clock down the layers of the timing network by means of a modified Precision Time Protocol (PTP) and synchronous GigaBit Ethernet.

Management Master

In the Management Master, all administrative tasks of the timing network are concentrated. There are multiple services running in this unit, allowing it to carry out the following functions:

- Dynamic Host Configuration Protocol (DHCP) Server
- Rapid Spanning Tree Protocol (RSTP) Root
- Simple Network Management Protocol (SNMP) Master

A standard DHCP server process is used to assign IP addresses to all nodes present in the timing network.

The SNMP Master is used to poll network switches for information about their current status in order to obtain a global status image of the timing network. It can also be used by administrators to configure said switches.

Since the FAIR Timing Master will broadcast command messages, the RSTP protocol is used to make the routing information for the timing network loop free. This is necessary because a loop in the routing tables would immediately lead to an infinite generation of network traffic.

Data Master

The Data Master for the FAIR accelerator deals with various different tasks. It possesses a high end CPU running an OS for easy interfacing to the control system, compatibility to FAIRs standard libraries and raw processing power as well as a Field Programmable Gate Array (FPGA) for parallelism, deterministic behavior and ultra low IO latency.

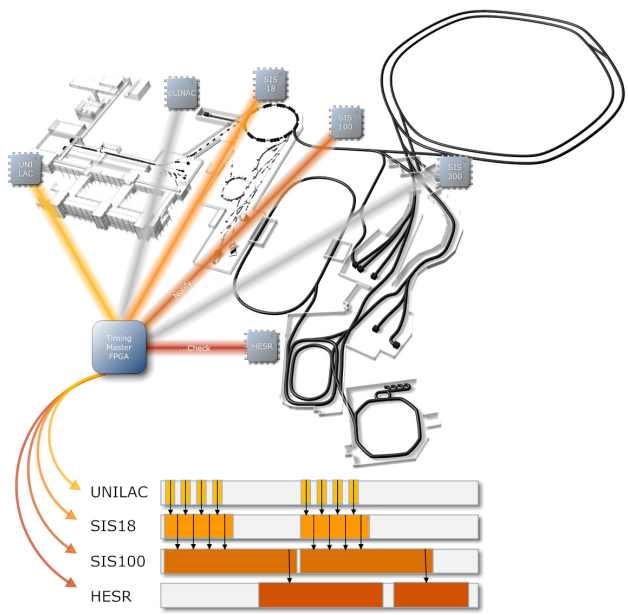

Figure 1: Sequences in a production chain [2].

The Data Master itself subdivided into three parts:

CPU/API Block Its CPU is fed by the LHC Software Architecture (LSA) with machine parameters derived from

physical requirements for beam production chains. These parameters are converted into sequence programs (Fig. 1), compiled and uploaded to the FPGA of the Data master.

FPGA/SoftCPU Cluster These programs are run in parallel on SoftCPU macros residing in the FPGA. They deal with sending out machine events paired with an execution time to WR nodes, reacting to interlocks and mutual synchronisation. At the moment, 32 of these Soft CPUs are foreseen in the Data Master, able to carry out 32 tasks full parallel with IO service times of less than 50ns.

FPGA/Event Concentrator An event concentrator macro in the data master will act as a bridge to the WR network. Its primary functions are aggregation of events into Ethernet Frames and to schedule transmission of these event messages over the timing network so they arrive on time at the respective nodes.

DESIGN CONSIDERATIONS

Why not just a High End CPU?

Modern multi-core CPUs are unsuitable because they cannot they be made deterministic enough for our purpose. This is not inherent in their design but stems from the fact that they need a full blown OS to use most of their features, not to mention RAM management. They also have comparatively slow IO handling. Even with high clock rates and vast amount of processing power at their disposal, their Interrupt Service Request (ISR) times are in the low millisecond range plus additional penalties from IO resource arbitration. While there are approaches which enforce a task based scheduling to introduce determinism, they still suffer heavily from the IO bottleneck of the carrier board.

High end multicore CPU

+ Performance clock for clock
+ Clock rates
– Determinism with (needed) OS
  – multithread scheduler
  – indirect memory management
  – garbage collection
– ISR times range in milliseconds
– IO Bottleneck

➠ Not suitable

Why no Embedded CPU with a Real Time Operating System?

While being considerably faster and more deterministic at IO and interrupt service times than high end CPUs, the ISR of embedded CPUs with real-time OSs still jitter in the microsecond range and show considerable lag under load, as shown in Figure 1.

Table 1: RTOS Latency Measurement Results [3].

	Interrupt Latency (µs)		**Context Switching** (µs)	
	max	**avg ±**	**max**	**avg ±**
Idle System				
RTL	13.5	(1.7 ± 0.2)	33.1	(8.7 ± 0.5)
RTEMS1	14.9	(1.3 ± 0.1)	16.9	(2.3 ± 0.1)
RTEMS	15.1	(1.3 ± 0.1)	16.4	(2.2 ± 0.1)
vxWorks	13.1	(2.0 ± 0.2)	19.0	(3.1 ± 0.3)
Loaded System				
RTL	196.8	(2.1 ± 3.3)	193.9	(11.2 ± 4.5)
RTEMS1	19.2	(2.4 ± 1.7)	213.0	(10.4 ± 12.7)
RTEMS	20.5	(2.9 ± 1.8)	51.3	(3.7 ± 2.0)
vxWorks	25.2	(2.9 ± 1.5)	38.8	(9.5 ± 3.2)

Embedded CPU with Real time OS

+ Fast IO
+ More deterministic
– ISR jitter
– lag peaks under load
– scalability

➠ Not suitable

A multi MCU approach to decreases IOs per MCU to be serviced and processor load would suffer from poor possibilities for process synchronisation between MCUs and lag from resource arbitration.

Why not Pure HDL on FPGAs?

Programmable hardware was the next option to investigate. While being extremely fast and the only system group capable of massive parallel processing, there are also disadvantages to consider. The main problem with HDL modules is a distinct lack of flexibility, since all sequential behavior needs to implemented as state machines. Those can only change their state in reaction to a set of external signals and their own state. This would severely increase the effort for conversion of LSA output and also mean that on the fly changes of functionality are out of the question. However, there is another option when using HDL macros.

Pure HDL circuits

+ completely deterministic
+ ultra low IO latency
– interfacing
– flexibility
– versatility
– design effort for all scenarios

➠ Not suitable

Why SoftCPUs?

SoftCPUs, little blocks of Hardware Description Language (HDL) code resembling embedded CPUs. They can offer almost all of the convenience and flexibility of a real embedded CPU, while also being closely connected to other FPGA circuits with extremely low latency. If there is no necessity for a MMU or the intention to use standard libraries, this is the best compromise between flexibility and ultra low latency design. Multiple instances with their own dedicated memory can speed up the design even more.

SoftCPU Cluster

+ completely deterministic
+ ultra low IO latency
+ interfacing
+ flexibility
+ versatility

➠ Best candidate

ARCHITECTURE

Choice of SoftCPUs

In the course of our work, many SoftCPUs have been evaluated [4]. We decided against closed source variants to keep the possibility of extending the macro with custom instructions. There is about a dozen open source SoftCPUs available today. The main criteria were speed, footprint, availability of toolchains and community/developer support. Our choice was the Lattice Micro32, a 32 Bit RISC processor macro running at about 150 MHz. While being a complete RISC processor with instruction and data caches and an interrupt handler, it has a low FPGA footprint of about 2000 logic cells. Toolchains are available, including a GNU cross compiler for Ansi C. We have also added support for the GNU debugger via JTAG interface. An average modern PFGA can offer around 250000 LUTs, four times more than enough to instantiate a SoftCPU cluster with 32 modules. Internal FPGA memory is much more likely to be the limiting factor, though. High end FPGAs contain >2 MB of internal memory. Assuming 2.5 MB, each of the SoftCPUs would get around 80kB of memory. This is more than sufficient for caches and program execution. The GNU debugger claimed about 1 kB of memory on our LM32 testbed.

Using Multiple SoftCPUs

With a One-SoftCPU-per-task policy, their IO handlers can be blocking and are not forced to use interrupts, eliminating time consuming context changes. Each SoftCPU has its own little memory controller, which completely eliminates resource arbitration lag. For fastest synchronisation of their programs, an $n * n$ sync matrix will also be implemented.

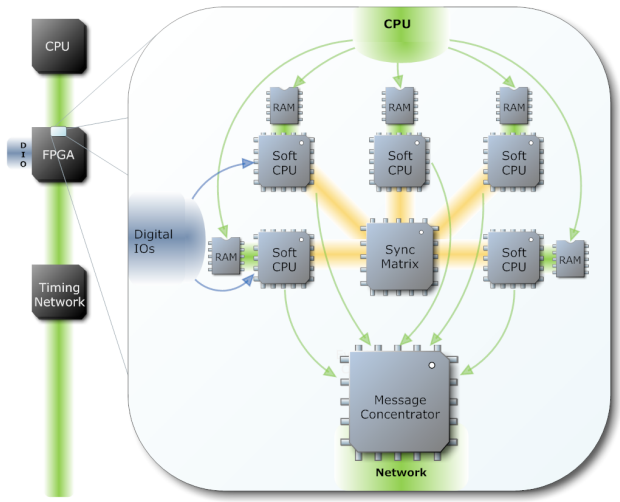

Figure 2: FPGA based SoftCPU Cluster [2].

CONCLUSION

After evaluation of timing requirements [2] and CPU, MCU and FPGA system properties, it showed that a mixed CPU/SoftCPU approach for the Data Master would be the best choice for FAIR. It is a very fast, flexible, scalable, easily extendable and future proof solution.

OUTLOOK

In the course of 2012, the design will be subjected to a real world test scenario. A first prototype of the FAIR Timing Master will be used in the testing and commissioning of the first machine module to be deployed for FAIR, the antiproton linear accelerator. Productive systems are planned to be put into service at GSI/FAIR in 2016.

REFERENCES

[1] P. Moreira, J. Serrano, T. Wlostowski, P. Loschmidt, G. Gaderer, "White Rabbit: Sub-Nanosecond Timing Distribution over Ethernet", IEEE Precision Clock Synchronization for Measurement, Control and Communication 2009, pp1-5, Brescia, October 2009.

[2] M. Kreider, "The FAIR Timing Master: A Discussion of Performance Requirements and Architectures for a High-precision Timing System", THCHMUST06, ICALEPS'11, Grenoble, France, 2011.

[3] T. Straumann, "Open source real-time operating systems overview", ICALEPS'01, San Jose, USA, 2001.

[4] W.W. Terpstra, "The Case for Soft-CPUs in Accelerator Control Systems", 2011, ICALEPS'11, THCHMUST05, ICALEPS'11, Grenoble, France, 2011.