# Clock control, interface and debug mechanism for an FPGA based x86 multi-processors emulator

Thorsten Mattner & Franz W. Olbrich
Intel Corporation
Theodor-Heuss-Strasse 7
D-38122 Braunschweig, Germany
thorsten.mattner@intel.com franz.olbrich@intel.com

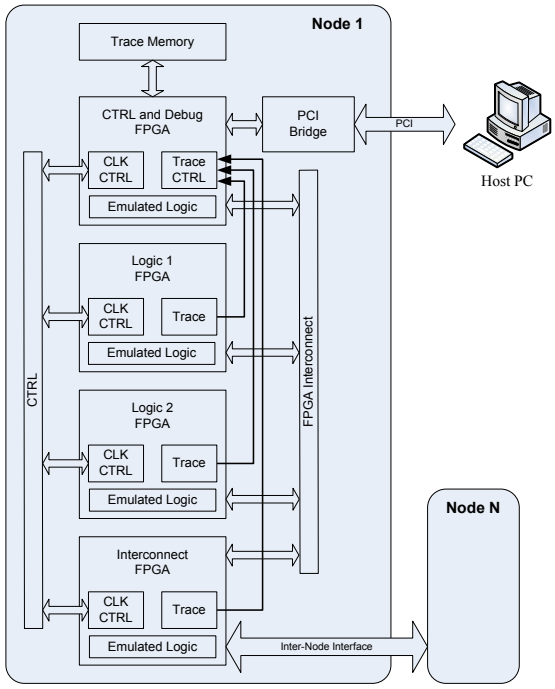
At the beginning of a development of future processors there is a need to simulate new architectural approaches to minimize risk of failure and to optimize the system performance. The simulation platform must be capable to handle various kinds of system hardware and software. The platform must also support functionalities like performance measurements, tracing for debug and hardware-software co-simulation. All this should be provided with a maximum performance. Simulation speed is the key factor to accomplish meaningful results in reasonable period of time. To execute detailed system performance analysis a large trace memory is required to store data during simulation.

Software simulation is a common way to provide this functionality but the simulation speed is too slow especially when evaluating a very complex system over a long time period for performance measurements. The achievable performance with simulation is in the region of several 100 Hz.

Another approach is using a hardware accelerator system. The hardware accelerator combines the flexibility of a software simulator with increased speed of the hardware acceleration. The disadvantages of this approach are the limited achievable gate-count and the still low simulation speed in the range of several 100 kHz.

An additional approach is the usage of FPGA prototyping systems available on the market. The reachable speed is higher as with the hardware accelerator. The achievable performance for this approach is in the range of a few MHz. The limitation of the flexible FPGA prototyping system is their generic topology. They are no optimized boards for a specific architecture available. The second limitation is that these boards do not provide tracing, control and debug features.

To overcome all the above mentioned problems we work on the development of a FPGA based x86 multi-processor emulator. The emulator system will include emulation clock control functionality, time-division-multiplexed interfaces and extendable debug and trace capabilities. Another advantage of the application specific implementation is that the architecture of the target system can already be considered during the planning phase to avoid possible bottlenecks regarding IO capabilities as they might occur in a flexible FPGA prototyping system which has to support a very flexible topology. The system performance will be in the range higher then 10 MHz.




The emulated system is running with one or more controllable emulation clocks which are derived from a global distributed low skew system clock. The emulation clocks can be enabled/disabled between two adjacent emulation clock cycles without loosing the clock accuracy in the system. This can either be initiated by a host system via software or based on some internal events of the system. After the event is handled the simulation can be continued without any restart. It is also possible to control the clock in a multi board setup of the system if a higher system complexity is required. The accurate clock controlling is the key feature to enable control and debug functionalities. The system is scalable via a high speed backplane and will fit into a compact PCI rack.

A basic functionality for every FPGA based emulation system is the time division multiplexing of physical interfaces (internal or device IO) over a high speed link between different FPGAs. This is especially needed if the complete system or certain sub-blocks doesn't fit into one FPGA and have to be partitioned over several FPGAs.

There will be a basic tracing functionality implemented to trace data without any impact on emulation speed. In addition extended trace

functionality provides tracing capabilities without any gap over a specific period of time. To support this higher tracing bandwidth the tracing logic is capable of stopping the emulator clock via the clock control logic whenever the internal 1 GB tracing buffer is running full. After the memory content was send to the host system the system continues to run.

To configure the system or extract the current state of internal logic the system will include configuration and status registers in every FPGA which can be programmed or read from a control and debug PC. These registers are automatically included into the FPGA via scripting based on configuration files. With this approach it is possible to access internal logic in the system to get relevant information about the current status. This approach also enables hardware-software co-simulation.