Sierra 2010 - Real-Time Kernel in Hardware

2010-03-18 Product Specification and Data Sheet

AGSTU AB

Address:	AGSTU AB, Dragverksgatan 138, S-724 74 Västerås, Sweden
Mail:	info@agstu.com
Phone:	+46 (0)70 66 89 517
Email:	info@agstu.com
URL:	www.agstu.com

Features

- Scheduling mechanism in hardware working in parallel to the CPU
- Accelerates the scheduling, with **no** scheduling, time tick, queue sorting etc. overhead
- 100 % deterministic
- Relieves pressure from the CPU
- Tasks, semaphores, flags, timers for delay and periodic tasks with deadline control etc.
- Handles smart and simple external interrupts as task with priorities (do not disturbed high priority tasks)
- Advanced hardware interface to start blocked task.
- Small software API, about 2Kb in memory
- Long Life HW Component, same function and time behaviour in the future.

Supported Devices

- Altera Family FPGAs
- Other programmable devices.

CORE Facts

Can be provided with Core	
Documentation	User's Reference Manual HW Reference Manual
Design File Formats	EDIF/ngc netlist; VHDL Source RTL
Verification	SW Test bench for HW/SW system (SW test program)
Instantiation templates	VHDL
Reference designs & application notes	Altera Reference Platform Tool
Additional Items	C-code API is delivered to utilize the functions of the Sierra RTK.
Simulation Tool Used	
Model Technology ModelSim™ SE/EE	
Support	
Support provided by AGSTU	

The Sierra Real-Time Kernel consists of:

- A priority driven scheduler with preemption
- A Resource Manager, which can be used for any kind of IPC, e.g. semaphores, flags
- A Time Manager, which contains functionality for handling delay, periodic start of tasks and deadline control.
- An Intelligent Interrupt Handler

This manual covers the use of HW based real-time kernel Sierra. Configuration of different Sierra, see web page www.agstu.com.

Tmq	Time Manager
Rmq	Resource Manger
IRQ	Interrupt Handler
GBI	Generic Bus Interface
TDBI	Technology Dependent Bus Interface

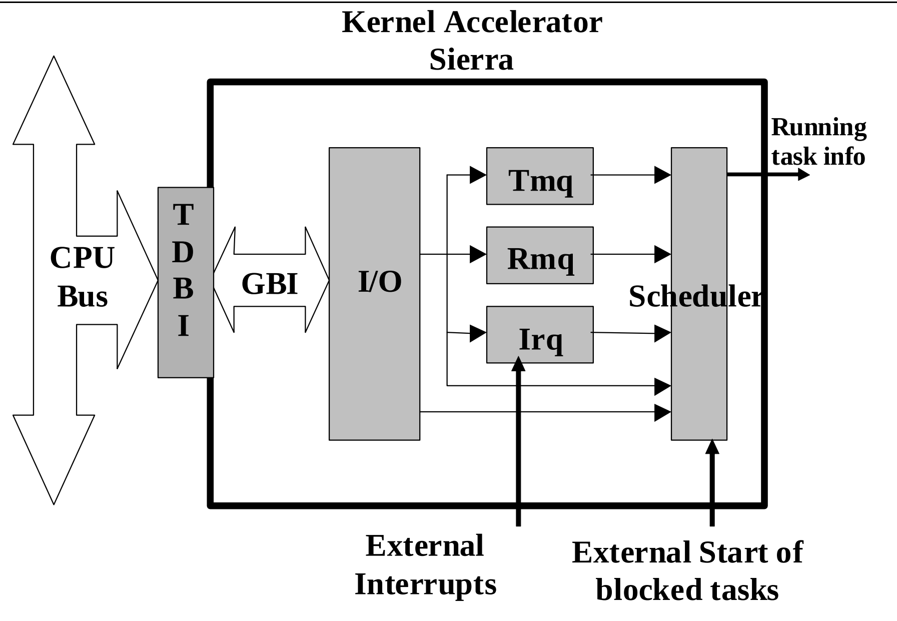

Figure 1 - Architecture of the Sierra

In general purpose OS´s time management control and all types of resource handling are time consuming due to large amount of queue handling. All types of queue handling, by that means searching in a specific queue, eat a lot of CPU cycles.

However, moving this to hardware much of the job can be performed in parallel to the CPU, also some new function as deadline control can be implemented in hardware. The Sierra has an easy and safe Interrupt Handler that makes interrupt handling much simpler. At the same time it improves performance and behavior.

Briefly, the external interrupts are routed to the Sierra and each interrupt service routine, ISR, is treated as a task waiting for an external event. When an interrupt occurs the ISR is scheduled as an ordinary task by the Sierra and will be started when it has the highest priority in the ready state.

Benefits

- When using the Sierra, there is almost no RTK-footprint in memory. Merely a driver to communicate with the hardware kernel
- Less cache misses, in systems with a cache, since the Sierra driver has such a small memory footprint.
- Optimized system behavior as the CPU-load is minimized
- Complete system predictability since hardware is 100 % deterministic and deadline control.
- Plug in architecture; add new functions to Sierra when needed.

Application

The Sierra solution is suitable for all kinds of small embedded systems, System-On-Chip (SoC) for example, and real-time systems in general to improve performance and predictability. Sierra can be used as a standalone real-time kernel or real-time operating system accelerator.

Specifics

In this version, Sierra supports 8 tasks at 8 priority levels, 8 semaphores, 8 flags, and two external interrupts.

On request, the Sierra can be extended to other configurations, e.g. more tasks, resources etc. Contact AGSTU for more information.

Functional description

The core is partitioned into modules as shown in and described in the text below.

Core Engine

The Sierra is partitioned into functional units

- Sierra Interface
- Scheduler
- Interrupt Handler
- Resource Manager; can be used for semaphores and flags
- Time Manager

Sierra interface

The interface to the Sierra is divided into a generic bus interface, GBI, and a technology dependent bus interface, TDBI. The GBI is bus independent while the TDBI is dependent on the specific bus in the system.

This design of the Sierra makes it very easy to interface it towards different kinds of busses.

The Sierra can be delivered with TDBI for Altera. The Sierra edif/ngc netlist has the GBI allowing the user to attach desired TDBI. The TDBI is delivered as source code. A customer can also develop other specific TDBIs or let AGSTU do it for a reasonable fee.

All communication with the Sierra is carried out through a set of registers as service calls. The service calls are decoded in the Sierra interface and routed to the specific unit that will handle the service call.

Scheduler

The Scheduler controls all scheduling in the Sierra. The number of tasks that the Sierra can handle is 16 and there are 8 priority levels for the tasks.

Normally a number of tasks are created when the system is initialized. However, the Sierra allows tasks to be created and deleted dynamically during runtime. When a task is created it is initialized to a specified state (blocked or ready). Tasks must have a priority and the priority must be initialized when the task is created. The scheduler guarantees that the task with the highest priority in the system that is ready. The scheduler uses a FIFO scheme for tasks on same priority level.

The Sierra supports the following task management primitives:

- Create/Delete task
- Block/Unblock task
- Enable/disable context switch
- Get task status

Resource Manager

The Sierra provides 16 semaphores and 4 flags. Each of these can be used for any form of interprocess communication, IPC, e.g. protecting of shared memory etc.

The following synchronization primitives are supported:

- Set flag/Clear flag/Wait for flag(s)
- Take/release semaphore

Time Manager

There are two time handling functions implemented, delay and support for periodic tasks. In software implementation, these functions will increase the CPU load.

In ordinary software RTOS the principle for the delay functionality is as follows. Every clock tick the RTOS has to check the delay queue and decrease each tasks timer and examine if any timer has expired. When a task timer expires the task has to be scheduled by the system again. All this work has a cost in time and this time increases with number of tasks using the delay function.

When it comes to the Sierra, all handling with timers etc. is done in hardware and all cost in time have been removed from the system.

The Sierra supports following time management primitives:

- Set/get time base
- Initialize periodic time for a periodic task
- Wait for next period with deadline control
- Delay

Interrupt Handler

The response time for interrupts generated by external devices must be kept short in all kinds of systems to obtain successful interaction with the external environment. Normally, external interrupts are routed to the CPU, which means that when an external interrupt occurs, the executing task will be interrupted. This has of course effects on the predictability of the system, as an ISR (Interrupt Service Routine) can preempt and run before a high priority scheduled task.

When using the hardware Interrupt Handler, external interrupts are routed in to the Sierra. Each ISR is treated as a task waiting for an external event. When an interrupt occurs the ISR is scheduled by the HW-Sierra and will start running when it has the highest priority in the ready queue. No SW execution at al.

The following primitives for interrupt handling exist:

- Wait for interrupt

Pinout

The generic bus interface, GBI, signals are described in figure 2 and table 2 below.

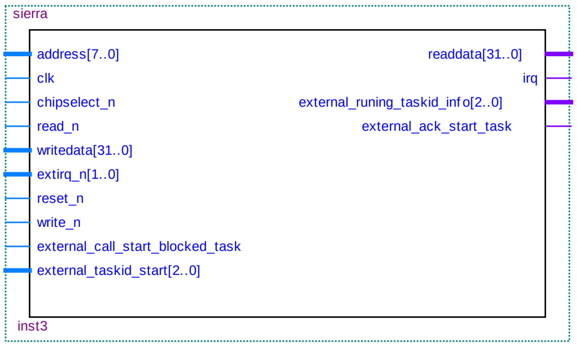

Figure 2: Sierra Generic Bus Interface (GBI)

Pin Name	Direction	Description
clk	Input, Sys	System clock
reset_n	Input, Sys	HW reset
cs_n	Input, Bus	Chip select
write_n	Input, Bus	Read / Write
addr(7:0)	Input, Bus	Address bus
din(31:0)	Input, Bus	Data bus in
dout(31:0)	Output, Bus	Data bus out

irq_n	Output, CPU	Task switch interrupt
extirq_n(1:0)	Input, User	External interrupts
External_runing_taskid_info[2..0]	Output, User	Updating Running task id (binary)
external_call_start_blocked_task	Input, User	Start of Blocked Tasks, Not used = ‘0’.
external_ack_start_task	Output, User	Start of Blocked Tasks
external_taskid_start[2..0]	Input, User	Start of Blocked Tasks

Table 2: Sierra Pinout

Verification Methods

Functional simulation of the Sierra carried out using Simulation with testbenches and HW/SW test.

The core has also been verified and used in systems with Altera FPGAs.

Functional co-verification of the Sierra kernel hardware together with the software API has been executed in Altera Embedded Development Kit.

Recommended design experience

Implementer of this core should be familiar with HDL design and design flows. Some knowledge in real time SW programming help to understand how this core works. Experience in designing systems with CPUs and other devices are recommended. This core can easily be integrated in any kind of system that has a CPU.
AGSTU AB also arrange intensive courses to learn FPGA SOC with Sierra. For more information go to agstu.com.

Design Services

AGSTU also offers core integration, core customization and other design services.

Ordering Information

This product is available from AGSTU, under terms of the SignOnce IP License. See www.agstu.se for additional information about this product.

Address:	AGSTU AB, Dragverksgatan 138, S-724 74 Västerås, Sweden
Mail:	info@agstu.com
Phone:	+46 (0)70 66 89 517
Email:	info@agstu.com
URL:	www.agstu.com

AGSTU cores are purchased under a Licence Agreement, copies of which are available on request. AGSTU retains the right to make changes to these specifications at any time, without notice. All trademarks, registered trademarks, or service marks are the property of their respective owners.

Related Information

Altera and Xilinx Programmable Logic

For information on programmable logic or development system software, contact your local Altera or Xilinx sales office, or visit www.altera.com or www.xilinx.com.