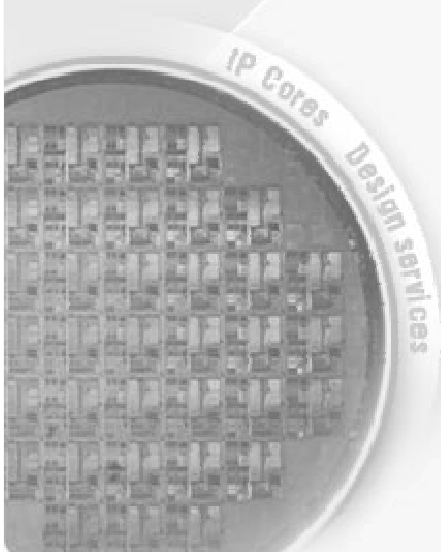




Embedded Software Development Using an Interpretive Instruction Set Simulator

Wojciech Sakowski – Institute of Electronics,
Silesian University of Technology,
Gliwice, Poland

Lukasz Mirek, Filip Rak – Evatronix SA, Bielsko-Biala,
Poland



evatronix

Outline

- ▶ The necessity of Instruction Set Simulator (ISS) implementation
 - Simulation approaches so far
 - ISS benefits
- ▶ R8051XC Instruction Set Simulator
 - Overview and model features
 - Architecture description
 - Architecture details
- ▶ Testing environment
 - Overview
 - Features
- ▶ Interfacing ISS with the TLM model
 - Overview and application
 - ISS vs. RTL IP component
- ▶ Conclusions

The necessity of Instruction Set Simulator implementation

- ▶ Simulation approaches so far
 - Native compilation
 - RTL model of the CPU
- ▶ The drawbacks
 - Native compilation – no architecture details given, no possibility for simulation of more than one thread at a time
 - RTL model of the CPU – it introduces overhead due to the CPU emulating itself

ISS benefits

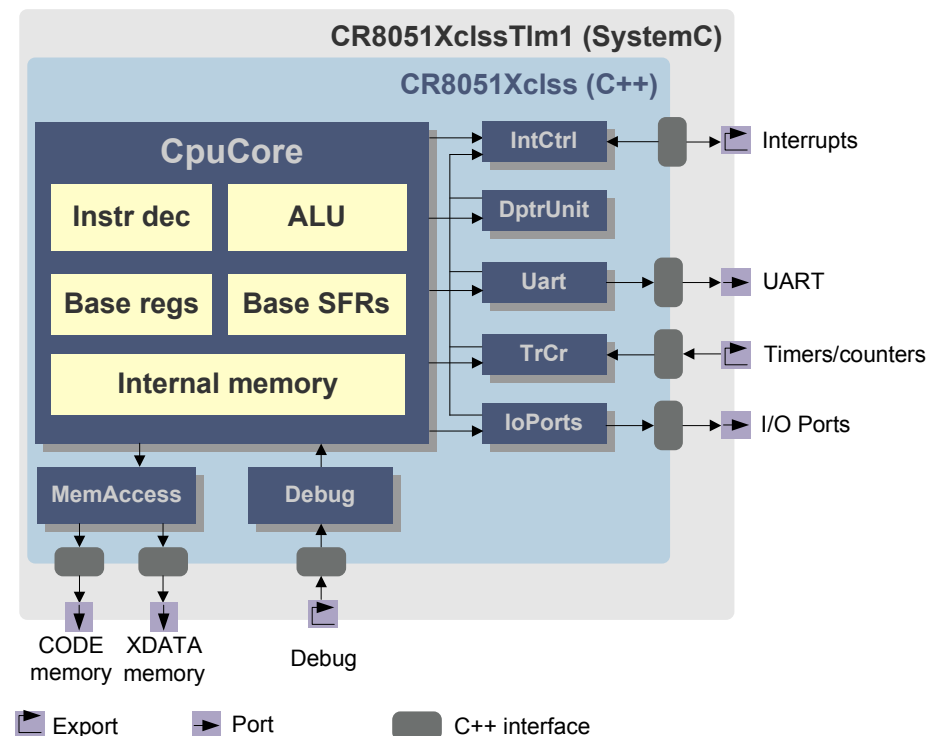
- ▶ Functional behavior identical to the RTL CPU model
- ▶ No internal architecture mapping
- ▶ 10.000 times better performance compared to RTL

R8051XC ISS – overview and model features

- ▶ Transactional model of the Evatronix R8051XC IP core in its full configuration
- ▶ Designed using C++ with external SystemC TLM wrappers (using TLM 2 library)
- ▶ ISS-specific assumptions:
 - Instruction-accurate functionality with no internal microarchitecture
 - Pure C++ using only generic types to improve simulation speed
 - Set of TLM wrappers to use ISS in SystemC environment
 - Static design with no internal processes
 - Look-up table instruction decoder, optimized for high-speed opcode processing
 - Either no time dependencies (pure PV model) or annotated timing (PV+T) with instruction-level time granularity
 - Standard communication interfaces (like TLM 2 for accessing external memories)

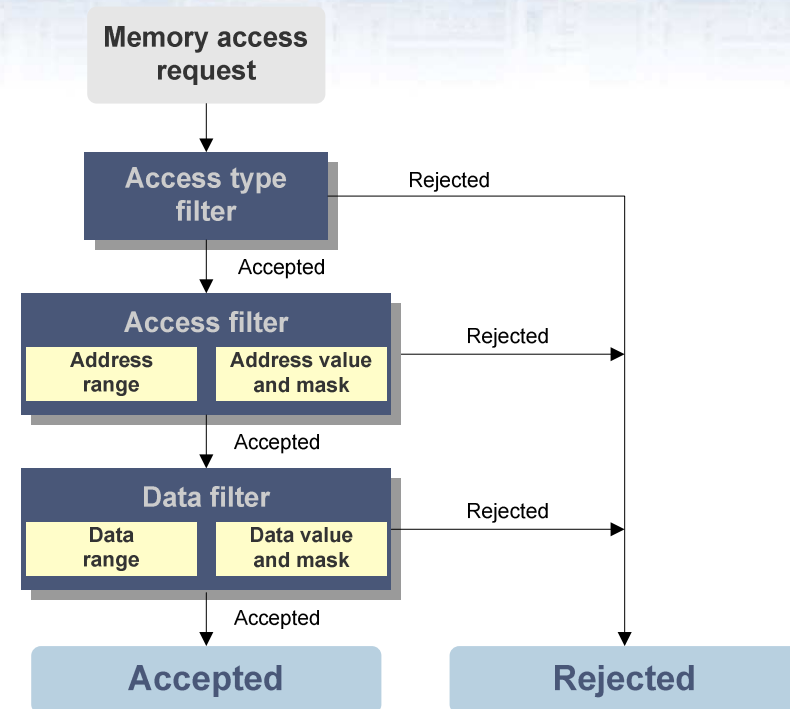
R8051XC ISS – architecture description

- ▶ The IIS is divided into following modules:
 - **CpuCore** - main CPU functionality with instruction processing unit and internal data memory implementation
 - **IntCtrl** – handling incoming interrupt processes
 - **DptrUnit** – Data Pointer register submodule
 - **Uart** – implements serial unit, present in 8051 CPU
 - **TrCr** – contains two timers and counters (T0 and T1)
 - **IoPorts** – provides access through external I/O ports
 - **MemAccess** - unified memory port independent of external interfaces
 - **Debug** - implements whole functionality related to flow control and accessing CPU internal resources



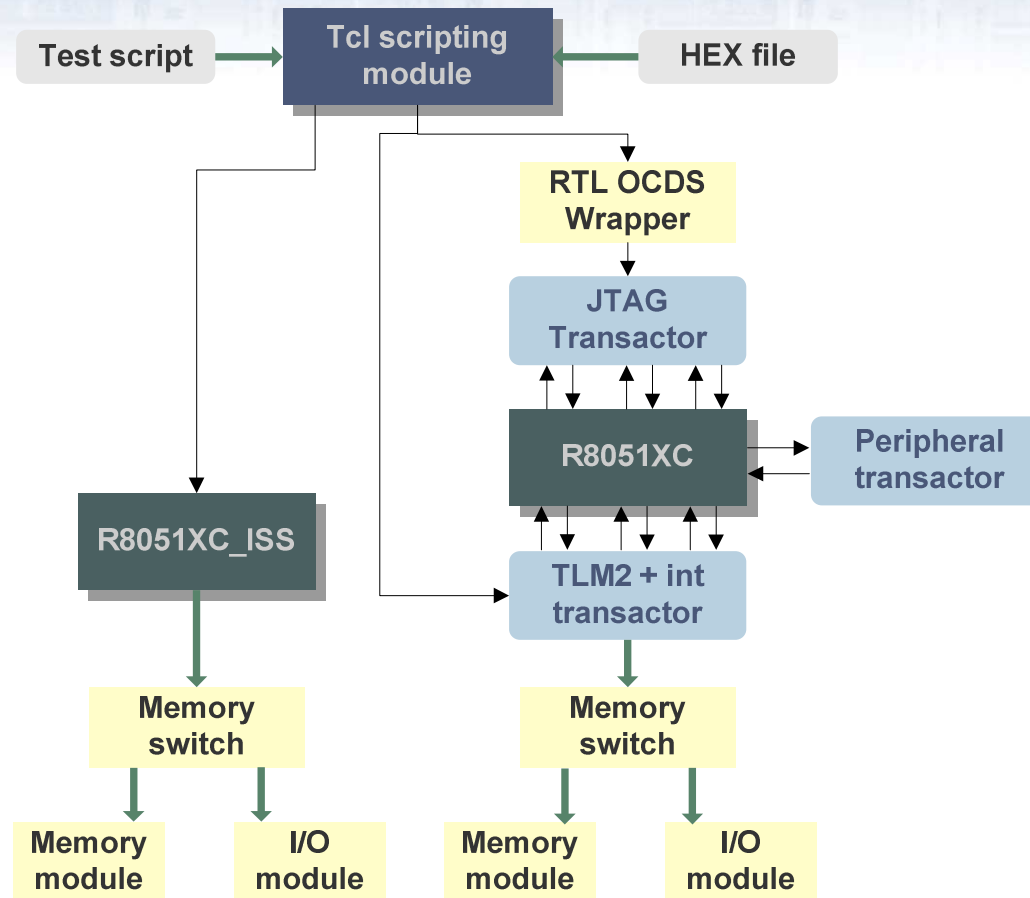
R8051XC ISS – architecture details

- ▶ **Instruction decoder:**
 - Instruction look-up table
 - Processing method accessible by indexing
- ▶ **External memory access:**
 - TLM2 external interfaces (sockets and generic payload)
 - TLM1 and pure C++ interfaces also available
 - DMI (Direct Memory Interface) usage (optional)
 - Hardware breakpoints logic
- ▶ **Data access breakpoints:**
 - Unified breakpoint logic for CODE, XDATA and IDATA space
 - Three operations monitored:
 - Read
 - Write
 - Code fetch (only for CODE)
 - Access address condition
 - Data condition
- ▶ **External SFR interface:**
 - Implemented as delegates array
 - Separate array for SFR read and write delegates



Data breakpoint algorithm

Testing environment - overview



Testing environment - features

▶ TCL console

- Tcl as a powerful scripting engine to control CPU
- Delegate-based Tcl <-> C++ / SystemC binding
- Debug_if routines accessible from Tcl script
- Built-in HEX parser

▶ OCDS functionality

- Nexus(TM)-compatible wrapper for debug_if
- Two-level access:
 - Nexus registers
 - JTAG signals

Interfacing ISS with the TLM model

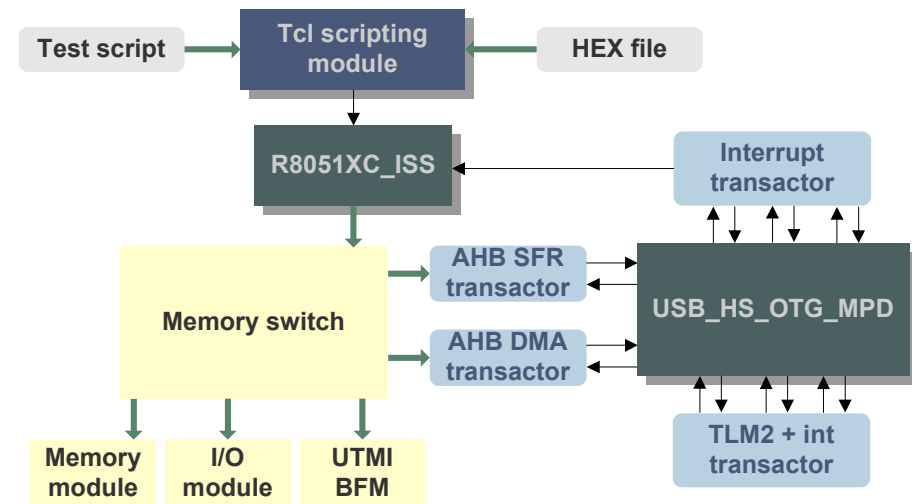
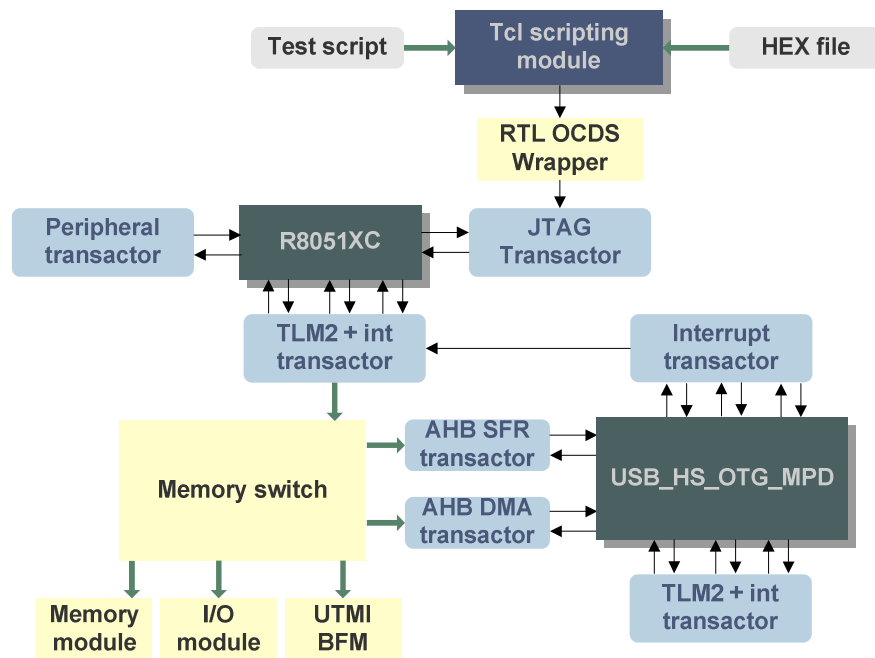
► Overview

- USB communication system
- TLM environment for high simulation speed
- Easy R8051XC <-> ISS exchange
- Communication via transactors

► Application

- UTMI BFM to simulate external bus devices
- Simple tests with some control / bulk / iso transfers
- Simulation speed improvement with ISS

Interfacing ISS with the TLM model – ISS vs. RTL IP component



Conclusions

- ▶ Very high simulation speed
- ▶ Easy CPU control
- ▶ Easy RTL <-> TLM exchange
- ▶ ISS – perfect solution for software developing, verification and profiling

Contact

www.evatronix.pl

Evatronix SA

Electronic Design Department,
Dubois 16, PL 44-100 Gliwice, POLAND
ipcenter@evatronix.pl

