1. Develop Base Stations
Radio Base Stations

Ericsson base stations support all major 3GPP and 3GPP2 technology tracks:
- GSM/EDGE
- WCDMA/HSPA
- CDMA
- LTE
Radio Base Stations

Macro  Main-Remote  AIR
A look inside

- Usually with the term system or embedded system we refer to hardware and software.
Develop base stations

- System-On-Chip
  Baseband, Radio, and Control SoCs

- System-On-Board
  Radio Unit and Digital Unit

- System-In-Cabinet
  RBS 6000 series of multi-standard base stations
System Design Process

THE DESIGN PROCESS AS A FLOW OF ACTIVITIES

System
- Hardware
- Software

TLM
- Architect System
- HW IP cores D&V
- HW Chip D&V
- HW Board D&V
- SW TOOLS
- Platform & App

IP

© 2014 Ericsson AB March 3, 2014
TLM advantages

Primary:
- Early Hardware Verification Environment
- Early Software Development and System Verification
- Enables early and massive System Exploration
- Enables early System Dimensioning

Secondary:
- IEEE Standard
- C++ offers a extensive code base
- Speed of Development
- Simulation and Verification Speed
Ericsson AB TLM

Community
- Ericsson joined Accellera as associate corporate level during 2012
- Ericsson became corporate level member 2014

Co-operating across projects and organization
- TLM steering group meetings every second week

Projects
- Methodology project – Apollo 2010
- Pilot project TLM for early SW development – Helios 2010
- Pilot project TLM for Architectural exploration – Vulcan 2011
- Sharp project TLM for early SW development – Ghost 2012
- Sharp project TLM for early HW verification – Atom 2012

Master thesis
- Accuracy of AT TLM models. How to compare RTL and TLM.
- TLM for virtual platforms
- TLM for verification
- SW statistics collection from TLM for HW exploration and dimensioning
2. SW Development
SW users at Ericsson

- Ericsson uses a TLM-2.0 LT based virtual platform for SW development
  - Hundreds of users
  - SW operates many months before chip and board is ready
  - Replaced legacy Virtual platform
SW Development

- Develop SW and run regressions
  - LT to get the speed
  - Memory/register accurate
- SW users operate at chip and board-level
- Standard debug tools are used independent if target is TLM or HW.
- More visibility with tools from EDA vendors.
System virtualization platform, single ASIC

- Test framework support
- 3rd party Debug tool support
- DspTools support
- (Ericsson internal)
- Debugger
- MDI
- DspTools Core
- QSim
- Legacy
- Virtual platform
- SVPIF client
- TCP/IP socket
- Test framework support
- In-house CCI framework

SVP

ASIC (TLM2)

- Accelerators
- Interfaces
- Control
- DSP cluster
- I/O
- Adapter
- SVPIF server
- Debug
- Attribute
- Execute
- Factory

Top

TCP/IP socket
System virtualization platform, board

- **SVP**
  - **Board**
    - CPU
    - TLM2 IP
    - FPGA (TLM2)
    - ASICS (TLM2)
  - **Adapter**
    - I/O
  - **Adapter**
    - I/O
  - **TCP/IP socket**
  - **3rd party TLM2 Simulator support**
  - **Test framework support**

- **In-house CCI framework**
  - SVPIF server
  - Debug
  - Attribute
  - Execute
  - Factory
In house CCI Module

- Must include header file
  ```cpp
  #include "svp.h"
  ```
- Should use svp_module
  ```cpp
  class Mymodule : 
  public svp_module {
  public:
    Mymodule(svp_modulename nm) : 
    svp_module(nm) 
    {} 
  }
  ```
- Otherwise regular sc_modules
- Must comply with modeling guidelines
In house CCI Attributes

- Store configuration data and state
- Are data members of SVP modules
- Can be used like regular data members inside the module
- Accessible from outside the class
- Two ways to deploy
  - Use ATTRIBUTE macro (preferred)
  - Explicitly instantiate attribute class

Internally attributes are linked to AttributeContainers
- Used to form attribute trees
- Orthogonal to sc hierarchy
- svp modules are attribute containers

```cpp
class my_module : public svp_module {
    my_module(svp_module_name smn)
    : svp_module(smn), a (123)
    { b = 456; }

    SVP_BASIC_ATTRIBUTE(a, uint32_t);
    SVP_BASIC_ATTRIBUTE(b, uint32_t);
};
```
In house CCI registers

- Store SW visible data and state
- Are members of SVP modules
- Organized hierarchically
  - module.{bank.}register.field
- Support access restrictions (r/w/rw)
- Accessible from outside the class
  - by SVP handlers and tool clients
- Provide APIs for reset, field-wise access, ...
  - eases internal usage and access
  - full register or individual fields
- Optional creation from reg descriptions
  - SystemRDL, IP-XACT
  - Handy for large register files (100s of regs)

// could be generated (MACRO|TOOL)

```cpp
template <T> class RegXY
: public er_register<T> {
    er_register_field<T,sBit,eBit,RW> f1;
    er_register_field<T,sBit,eBit,RW> f2;
    myReg(RegisterName an, T resetVal)
    : er_register<T>(an, resetVal), f1("f1", this), ...
    {}
};

class my_module
: public svp_module {
    RegXY<uint32_t> myReg;
    // er_register<uint32_t> noFieldsReg;
    my_module(svp_module_name smn)
    : svp_module(smn), myReg("myReg", 0xffff)
    // , noFieldsReg("noFieldsReg", 0x0)
    {}
3. Performance evaluation
Performance evaluation

- TLM-2.0 AT models used for HW IP level performance exploration
- TLM-2.0 LT models used at system-level to acquire SW load models
Roadmap for architectural exploration

- It is not in the current roadmap to develop AT models of complete ASICs or boards
- AT modeling used at IP level or subsystem level to do exploration or dimensioning
- LT level have shown accuracy enough to be used for
  - System level exploration
  - To acquire SW load model that is used in dimensioning or exploration
Performance Evaluation

- Performance Evaluation is to quantify the services given by the system as a function of scenarios
  - Quantifying services = Metric or performance metric
  - Scenarios = Stimuli or load

Goals are:

- **Performance dimensioning**
  - Assure the selected system meets performance goals.
  - Metric high accuracy

- **Performance exploration**
  - Performance evaluation of multiple systems that are then compared.
  - Metric medium accuracy (Relative comparison)
SW load models acquired from TLM LT

- Use SVP to collect samples from SW applications.
  - Usage of buses
  - Usage of memories
  - Usage of HW cores
  - etc.

- Make statistical analysis of the samples

- Build a flexible TLM traffic generator targeting performance evaluation.

- Used also in static analysis
IP level exploration

AT Model vs. Original RTL

Later RTL vs. Original RTL after boosting

30% Performance boosting!
4. HW functional verification
HW Verification at Ericsson

Today…

- TLM-2.0 LT models are used as references at HW IP level
- TLM-2.0 LT models used as development platform for chip level verification
Chip level verification

1. Development phase
   - Test case SW
   - TLM

2. Regression phase
   - Test case SW
   - RTL
   - Emulator
   - Target

<table>
<thead>
<tr>
<th>Platform</th>
<th>Relative load time + execution time</th>
</tr>
</thead>
<tbody>
<tr>
<td>RTL</td>
<td>15000x (days)</td>
</tr>
<tr>
<td>HW emulation</td>
<td>80x (minutes)</td>
</tr>
<tr>
<td>TLM LT</td>
<td>1x (seconds)</td>
</tr>
</tbody>
</table>

### Advantages

- TLM available early and used as stable development platform for **SW** driven test cases.
- Develop and run design loop is fast.
TLM in chip level verification

- Software driven Verification
- Same verification platform for all abstractions, TLM, RTL, Emulation and HW.
- This setup allows us as well to verify and debug software.
- TLM model used for test software development in this setup
- The verification environment supports both test software development and design verification.
- Possibilities to run real software will improve our confidence and pave the wave for a smooth software integration.
- Regressions are run on all abstractions
HW IP verification

- **Top-down design approach**
  - TLM as DUT and reference REF
    - High abstraction level
  - DUT later exchanged to matured RTL
    - Low abstraction level
Challenges

- How smart should the scoreboard be to identify the mismatch type?

- Is the observability enough?
  - Internal state divergence due to different model accuracy
HW verification: LT or AT?

- How can we move the REF model to low abstraction level for achieving the verification goal?
- How can we maximize the model reusability?
**HW verification: reusable LT module**

- Drive TLM at critical condition.
  - Increase detail level with less modeling effort
- TLM behaves partly as a state observer and checker.
  - Increase observability for debugging

**LT functional prototype**

- **F(x)**
- **G(x,t)**
- **H(x)**

**Event driven LT verification mode**

- **F(x)**
- **G1(x)**
- **G2(x)**
- **H(x)**

**UVM control interface**

**Interoperability?**

**Develop time**

- For Timing
- For Function

**Abstraction Level**
Summarize LT and AT

- LT adds value to several user groups in HW and SW.
- Verification requires refined LT but not AT.
- LT models are useful for system exploration and to acquire SW load models.
- AT models used for IP level or subsystem level exploration and dimensioning.
Thank you