Development and verification of the TOTEM DAQ firmware

Adrian Fiergolski

Warsaw University of Technology, Poland
Italian Institute for Nuclear Research, Bari section, Italy

System Verilog & UVM workshop; CERN
14 November, 2013

1. Toolchain
2. Firmware development
3. Verification
4. Miscellaneous tools

Toolchain used in TOTEM

Synthesis:

- Synopsys Synplify Premier (*I-2013-09-1*)

Simulation:

- Mentor Graphics QuestaSim (*10.2c*)
- Mentor Verification IP (*10.2*)

Miscellaneous:

- HDLmake (*Open Hardware repository*)
- Natural Docs

SystemVerilog in writing synthesizable hardware models

Advantages of the System Verilog:

SystemVerilog in writing synthesizable hardware models

Advantages of the System Verilog:

- interfaces

  much more intelligent than VHDL records!

SystemVerilog in writing synthesizable hardware models

Advantages of the System Verilog:

- interfaces
  much more intelligent than VHDL records!
- syntax compactness of Verilog
  useful extensions: specialized procedural blocks, C like data types, typedef, structures, priority and unique decision modifiers...

SystemVerilog in writing synthesizable hardware models

Advantages of the System Verilog:

- interfaces
  much more intelligent than VHDL records!
- syntax compactness of Verilog
  useful extensions: specialized procedural blocks, C like data types, typedef, structures, priority and unique decision modifiers...
- easy integration wit simulation
  assertion checks & coverage blocks directly in hardware description, straightforward hierarchical paths to internal signals...

SystemVerilog in writing synthesizable hardware models

Advantages of the System Verilog:

- interfaces
  much more intelligent than VHDL records!
- syntax compactness of Verilog
  useful extensions: specialized procedural blocks, C like data types, typedef, structures, priority and unique decision modifiers...
- easy integration wit simulation
  assertion checks & coverage blocks directly in hardware description, straightforward hierarchical paths to internal signals...

Disadvantages of the System Verilog:

- danger to end-up with an architecture which doesn't follow author's intention
- EDA tools are still immature with respect to legacy languages

Immaturity of EDA tools supporting System Verilog

Issues noticed during everyday Synplify usage:

- poor constraint support
  - conversion between UCF and SDC, FDC
  - buses in FDC files
- problems connecting VHDL record-ports with System Verilog structures
- parametrized vector-ports (unbind variables)
- coverage groups are not skipped by compiler (workaround: *`ifdef MODEL_TECH*)

The tool support is available and bugs are gradually fixed.

Recomended bibliography

- **"SystemVerilog for Design"**
  Stuart Sutherland, Simon Davidmann, Peter Flake
  Springer, Jul 20, 2006

Verification flow

The main power of System Verilog in verifcaiton comes from:

- Universal Verification Methodology (UVM) $\rightarrow$ re-usability
- availability of EDA verification libraries providing support for common interfaces (Ethernet, PCIe, USB, I2C...)

Verification flow

The main power of System Verilog in verifcaiton comes from:

- Universal Verification Methodology (UVM) $\rightarrow$ re-usability
- availability of EDA verification libraries providing support for common interfaces (Ethernet, PCIe, USB, I2C...)

Verification methodology

Verification flow

The main power of System Verilog in verifcaiton comes from:

- Universal Verification Methodology (UVM) $\rightarrow$ re-usability
- availability of EDA verification libraries providing support for common interfaces (Ethernet, PCIe, USB, I2C...)

Verification methodology

- each FPGA design has its own test environment trying to cover all functionalities of the DUT by set of sequences

Verification flow

The main power of System Verilog in verifcaiton comes from:

- Universal Verification Methodology (UVM) $\rightarrow$ re-usability
- availability of EDA verification libraries providing support for common interfaces (Ethernet, PCIe, USB, I2C...)

Verification methodology

- each FPGA design has its own test environment trying to cover all functionalities of the DUT by set of sequences
- modules with weak accessibility from external interfaces of an FPGA have dedicated test environment

Verification flow

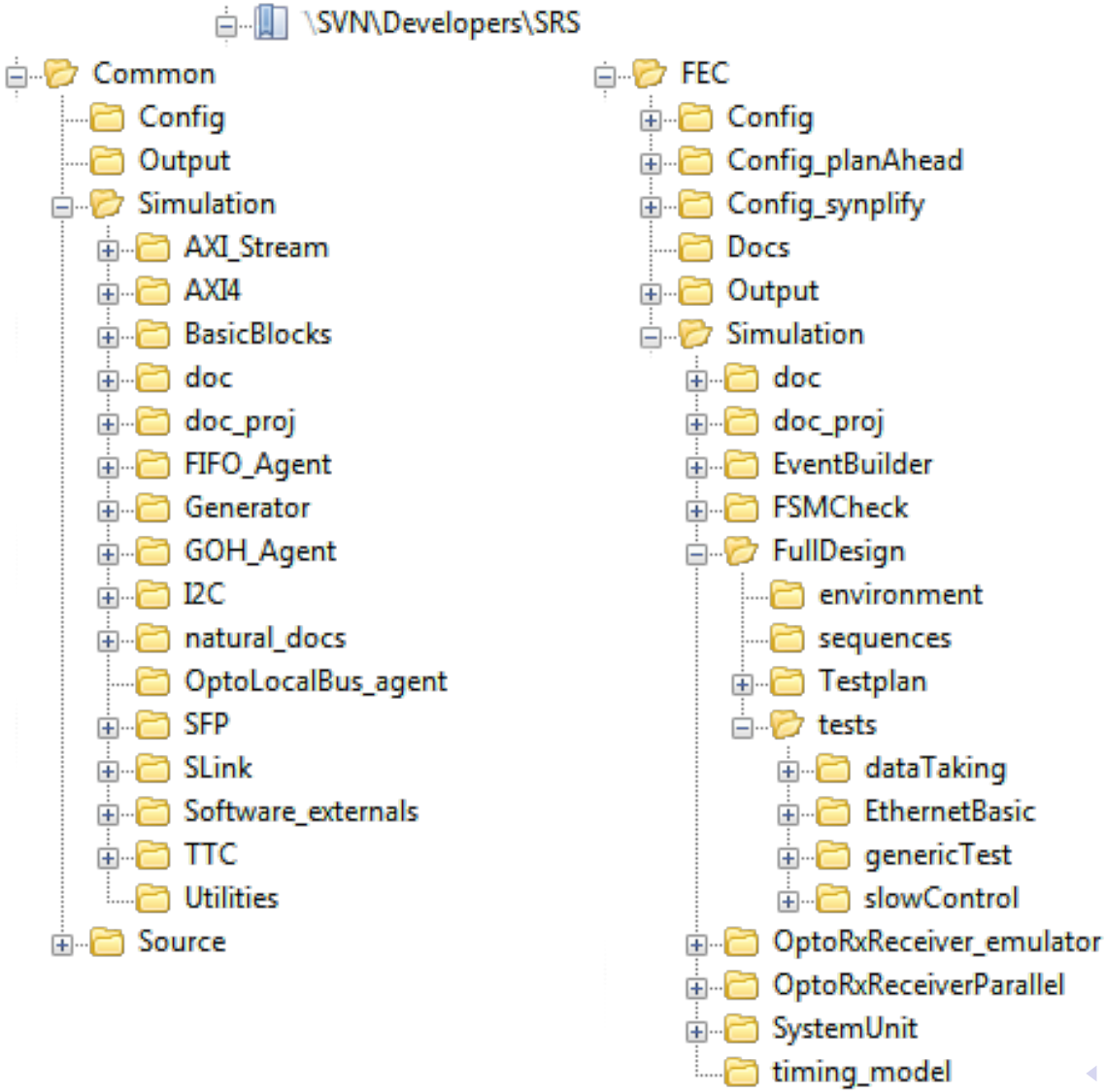
The main power of System Verilog in verifcaiton comes from:

- Universal Verification Methodology (UVM) → re-usability
- availability of EDA verification libraries providing support for common interfaces (Ethernet, PCIe, USB, I2C...)

Verification methodology

- each FPGA design has its own test environment trying to cover all functionalities of the DUT by set of sequences
- modules with weak accessibility from external interfaces of an FPGA have dedicated test environment
- the tests use Mentor's VIP library (*common interfaces*) and internally developed set of packages providing UVM agents (*custom interfaces*)

Testbench organization



Verification policy

- Adding new functionality, firmware has to pass set of general tests before proceeding to hardware verification

Verification policy

- Adding new functionality, firmware has to pass set of general tests before proceeding to hardware verification
- All agents come with coverage blocks allowing estimation of the verification progress

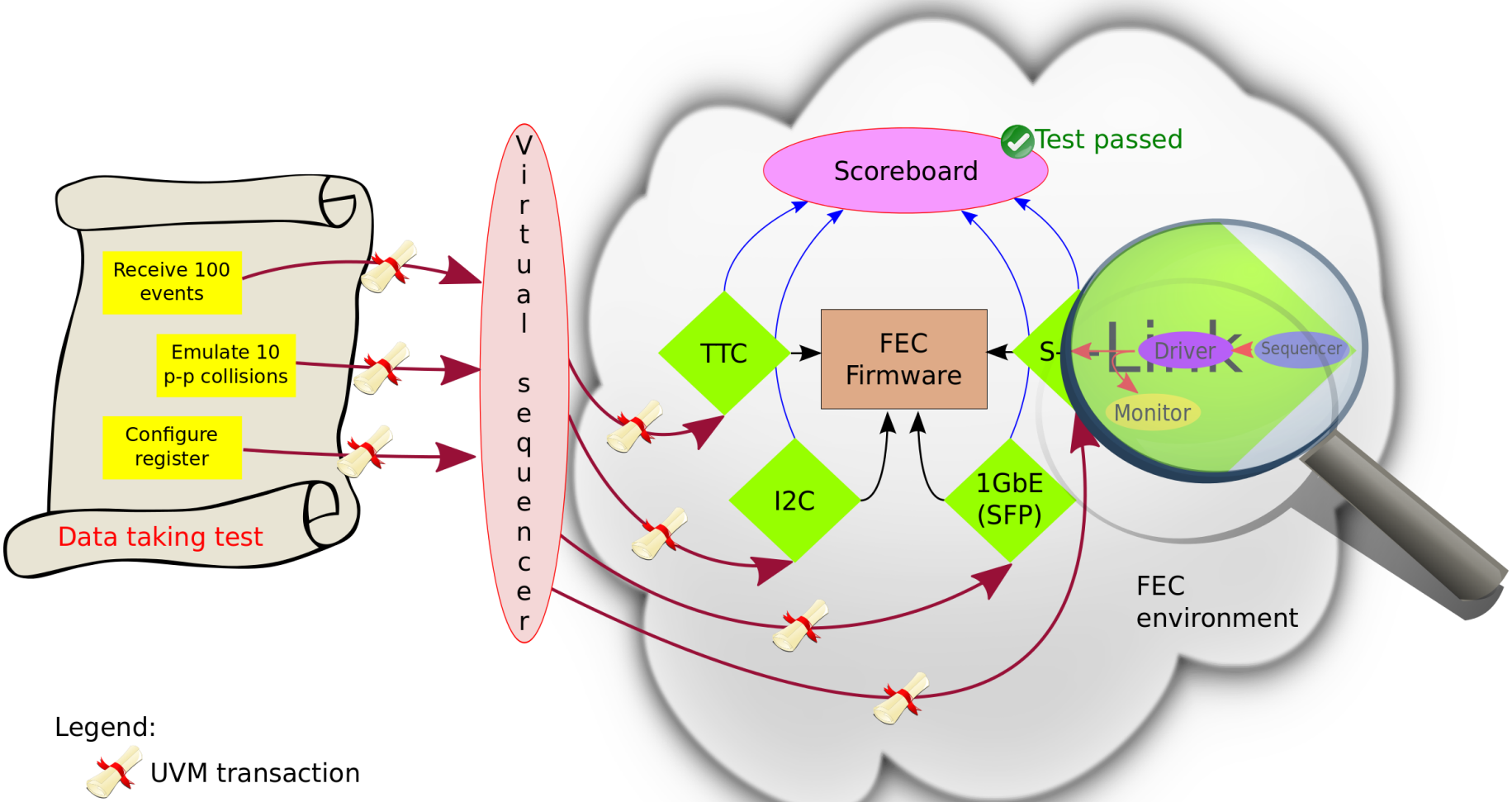
Verification policy

- Adding new functionality, firmware has to pass set of general tests before proceeding to hardware verification
- All agents come with coverage blocks allowing estimation of the verification progress
- rule of thumb $\rightarrow$ dedicated test environments for single modules are created if search for a bug takes more than one hour

Verification policy

- Adding new functionality, firmware has to pass set of general tests before proceeding to hardware verification
- All agents come with coverage blocks allowing estimation of the verification progress
- rule of thumb $\rightarrow$ dedicated test environments for single modules are created if search for a bug takes more than one hour
- all verification classes are code-documented following modified NaturalDocs scheme

Basic UVM testbench



Immaturity of EDA tools supporting System Verilog

Issues noticed during everyday Questa and VIP usage:

- mismatch in waveforms between -vopt and -novopt flag for vsim
- continuous assignment issue
- unstable UVM transaction recording
- bugs in VIP

The bugs are reported to Mentor Graphics and gradually fixed.

Recomended bibliography

- **"SystemVerilog for Verification : A Guide to Learning the Testbench Language Features"**
  Spear, Chris
  Dordrecht : Springer, 2008
- **Verification Academy**
- **UVM tutorial**

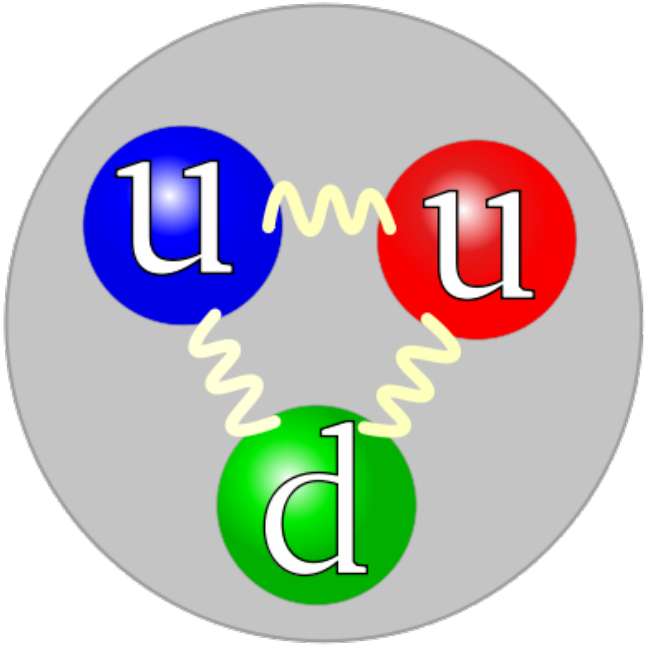
Miscellaneous tools

HDLmake

Python application automatizing makefile generation for verification. Using Manifest files describing content of separate packages, it is able to include files distributed over directory structures and add required compilation flags if needed.

IDesignSpec

It automatize register generation and maintenance. Properties of the registers (addresses, mode, kind,etc.) are described in user friendly Office file (Word, spreadsheet). Tool is able to generate synthesizable model of registers following specific common interface (AMBA-AHB, Avalon,Proprietary), UVM register model, C/C++ header.



Thank you for your attention.