DATASHEET

MERCURY COMPUTER SYSTEMS™

SVE - Simulation and Verification Environment

Simulation and Verification Tool for Mercury's FPGA Systems

- Simulates and verifies FPGA-based application designs
- Dramatically reduces time to market
- Supports all external board interfaces
- Integrates fabric interfaces (SRIO, PCIE, 10GE) seamlessly with unified easy-to-use APIs
- Domain knowledge-developed and deployed by system and application experts

SVE from Mercury Computer Systems is a complete end-to-end simulation and verification tool for Mercury FPGA-based systems, designed to enable application developers to quickly model and verify application logic, dramatically reducing time to market. The SVE architecture is based upon the OVM open-standard verification methodology (Open Verification Methodology, ovmworld.org). It is designed to allow users to quickly simulate and verify new designs, reducing verification time by up to 60-70% by providing complete verification infrastructure, integrated fabric, Bus Functional Models, and a library of pre-built tests with a unified easy-to-use API. This highly configurable, comprehensive solution is designed to test the FPGA and all its interfaces at the system level by emulating real-world traffic scenarios with a selectable degree of randomness.

SVE automatically creates a significant amount of the required fabric transactions and checks the responses. It utilizes coverage metrics to monitor progress against goals. By providing support for constrained random testing, it allows faster debugging of proprietary functionality, corner-case testing, and error detection.

Enhanced-Function Verification Solution Offers High Level of Abstraction

The architecture shown in Figure 1 allows the user to write test cases at a higher level of abstraction using an API, because SVE seamlessly translates test-case API commands into the underlying protocol. The same API is used by software engineers developing C code in the system, allowing software and hardware engineers to easily communicate. The API also allows the test writer to code tests at a higher level of abstraction and, therefore, to generate tests quickly, providing valuable coverage for the design.

Other enhanced features include:

- Compliance with industry-standard System Verilog-based OVM environment
- Concurrent operation of various number and types of interfaces with synchronization between them
- Programmable data/traffic generation and response checking
- Error insertion/detection
- Code and functional coverage as a measure of completeness
- Configurable symbol or bit serial interface
- Library of pre-built directed, constrained random, and performance test cases for testing of specific functionality and detecting corner cases
- Directed, constrained-random performance tests using a unified API
- Sparse memory for Bus Functional Models
- Deposit of data files in memories at time 0
- Serial interfaces that can run in parallel mode to improve simulation time
- Comprehensive documentation package

Productivity Enhancement

The SVE environment is designed to improve FPGA application developers' productivity by featuring software-aligned APIs that are uniform across protocols, and decoupled from their details. Although FPGA infrastructure complexity has increased dramatically due to the integration of switch fabric IP cores (such as SRIO, PCIE, and 10GigE), the test cases have fewer lines of code, are easier to create and debug, are portable between projects, resulting in a 60 – 70% productivity improvement.

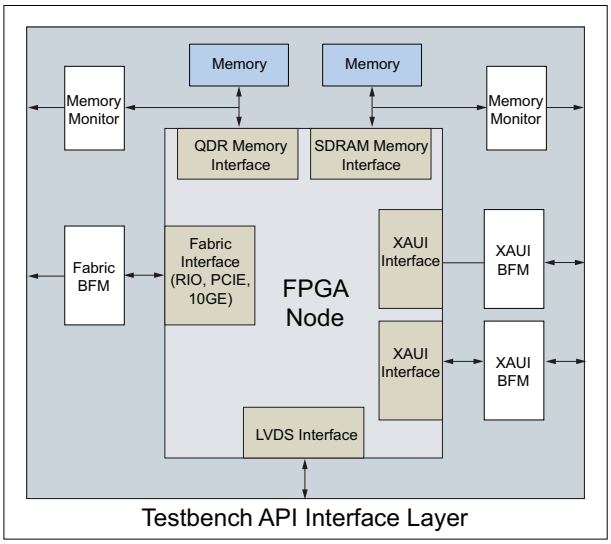

Figure 1. *SVE Verification IP architecture*

Sample SVE Verification IP Code

The following code example shows the simple actions of writing a data file to a memory connected to the FPGA, and then setting up a DMA transfer that transfers data to BFM memory and then displays and checks the data.

```
ovm_report_info(c_name,"Reading data from ascii file sample.txt ");
putfile(.adr(sram0_base_addr), .file_name("sample.txt"), .byte_cnt(256),
        .format(asciihex), .width_bytes(8), .proto(ep0));
dma_enable(.ep_icr_base(icr_base),.proto(ep0));
dx_create(.cp_strt_adr(cp_strt_adr),.cmd(NREAD),.local_adr(local_adr),
        .remote_adr(remote_adr1),.byte_cnt(byte_cnt),.sync(1),
        .next_cp(next_cp_adr(cp_base_adr, i+1)),.proto(ep0));
dx_send_value(.cp_strt_adr(cp_strt_adr),.local_adr(34'h0),
        .value(deflt_last_value),.remote_adr(deflt_last_adr),
        .proto(ep0));
ovm_report_info(c_name,"Start DMA transfer");
dx_start(
        .cp_strt_adr(cp_base_adr),.chan(CHAN0),
        .ep_icr_base(icr_base),.proto(ep0));
ovm_report_info(c_name,"Read and Dump BFM memory");
ioctl.op = RIO_RMEM;
getbuf(remote_adr2, rdat, byte_cnt, ep0, ioctl);
bufprint(rdat, byte_cnt, 8, ep0.big_endian);
getcheckbuf(remote_adr2, rdat, dat, byte_cnt, pass, ep0, ioctl);
```

Specifications

Language
- System Verilog HDL OVM

Supported platforms and simulators
- Mentor®

Supported switch fabrics
- PCI Express®, RapidIO®, XAUI, 10GE

Symbol interface 10 bit
1x, 2x and 4x SRIO serial interfaces
1x, 2x, 4x, 8x PCIe serial interfaces
4x XAUI and 10GE serial interfaces
Scoreboard port
Synchronization between various interfaces
Input/output and message-passing protocols
All transaction flows and priorities

Product Options

Single or multi-use license

Compliance

RapidIO specification, Revision 2.1
PCI Express specification, Revision 2

Prerequisite Product

Mentor Questa Sim Verilog/VHDL and Synplicity Verilog/VHDL
Avery PCIe BFM (only needed for PCIe designs)

1922.01E-1218-DS-sveip

MERCURY COMPUTER SYSTEMS™

INNOVATION THAT MATTERS™

CORPORATE HEADQUARTERS
50 Minuteman Road • Andover, MA 01810 USA
(978) 967-1401 • (866) 627-6951 • Fax (978) 256-3599

EUROPE MERCURY SYSTEMS, LTD
Unit 1 - Easter Park, Benyon Road, Silchester, Reading
RG7 2PQ United Kingdom
+ 44 0 1189 702050 • Fax + 44 0 1189 702321