



# Generating and Maintaining VMM-Compliant Verification Environments

by

Dr. Ambar Sarkar

Chief Verification Technologist

Paradigm Works Inc.

- Challenges of creating/maintaining VMM environments
- SystemVerilog FrameWorks™ Template Generator
- Generating verification environments
  - Standalone DUT
  - SOC DUT
- Conclusion

# Challenges of creating and maintaining VMM environments

---

- Overcoming initial learning curve
  - Working with teams with mixed skill-levels
- Following company specific requirements
  - Supporting common practice
  - Communicating guidelines and conventions across multiple sites
- Avoiding evils of cut and paste



# SystemVerilog FrameWorks™ Template Generator (SVF-TG)

---

Ambar Sarkar



4

- A tool to jumpstart a SystemVerilog testbench environment
  - Reduces team ramp-up time significantly
  - Helps and enforces adherence to VMM methodology
  - Captures verification expertise not easily accessible elsewhere
  - Promotes reuse and maintainability
- Proven on real projects across multiple clients



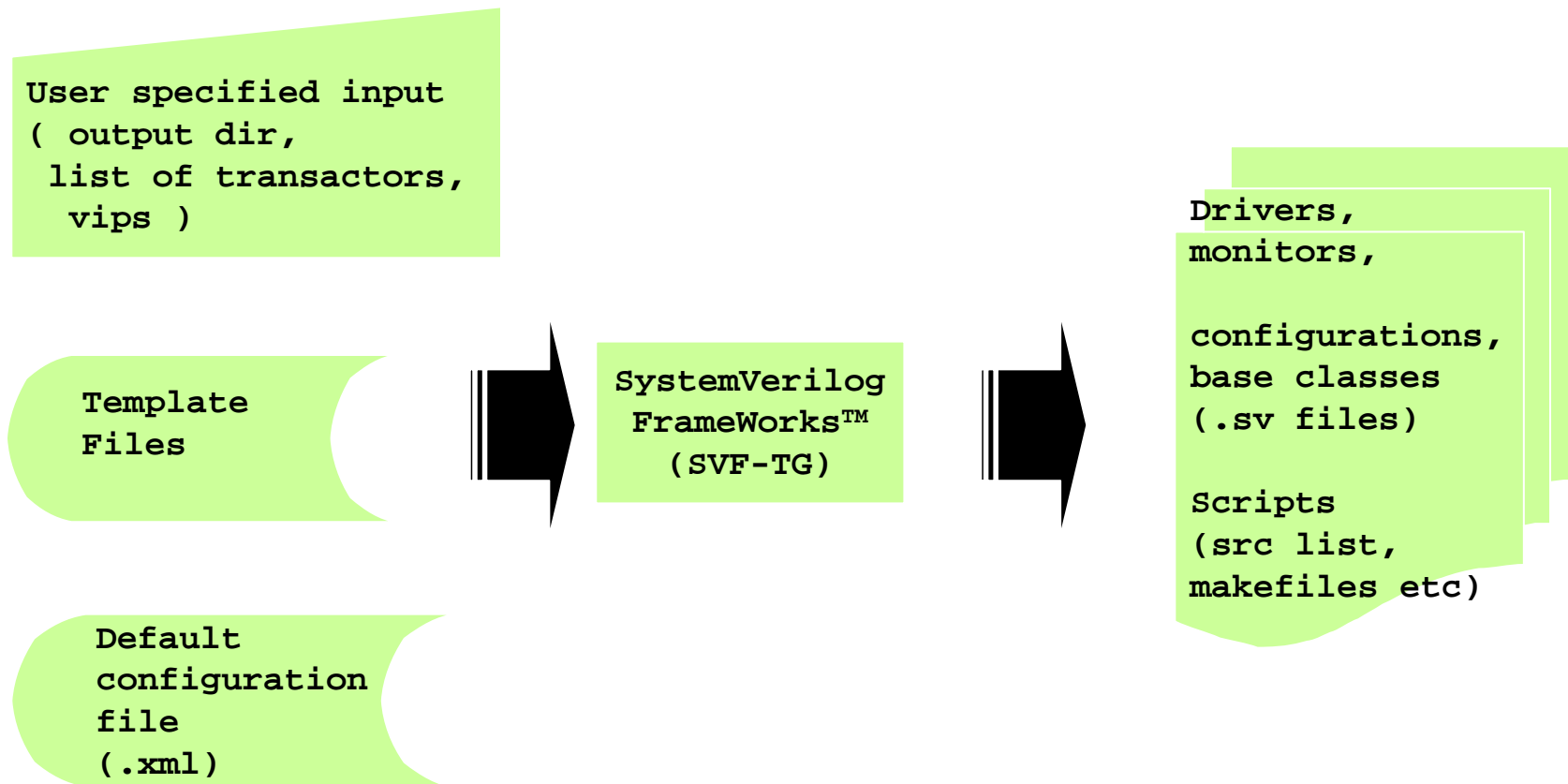
# System Verilog FrameWorks™ Template Generator (SVF-TG)

---

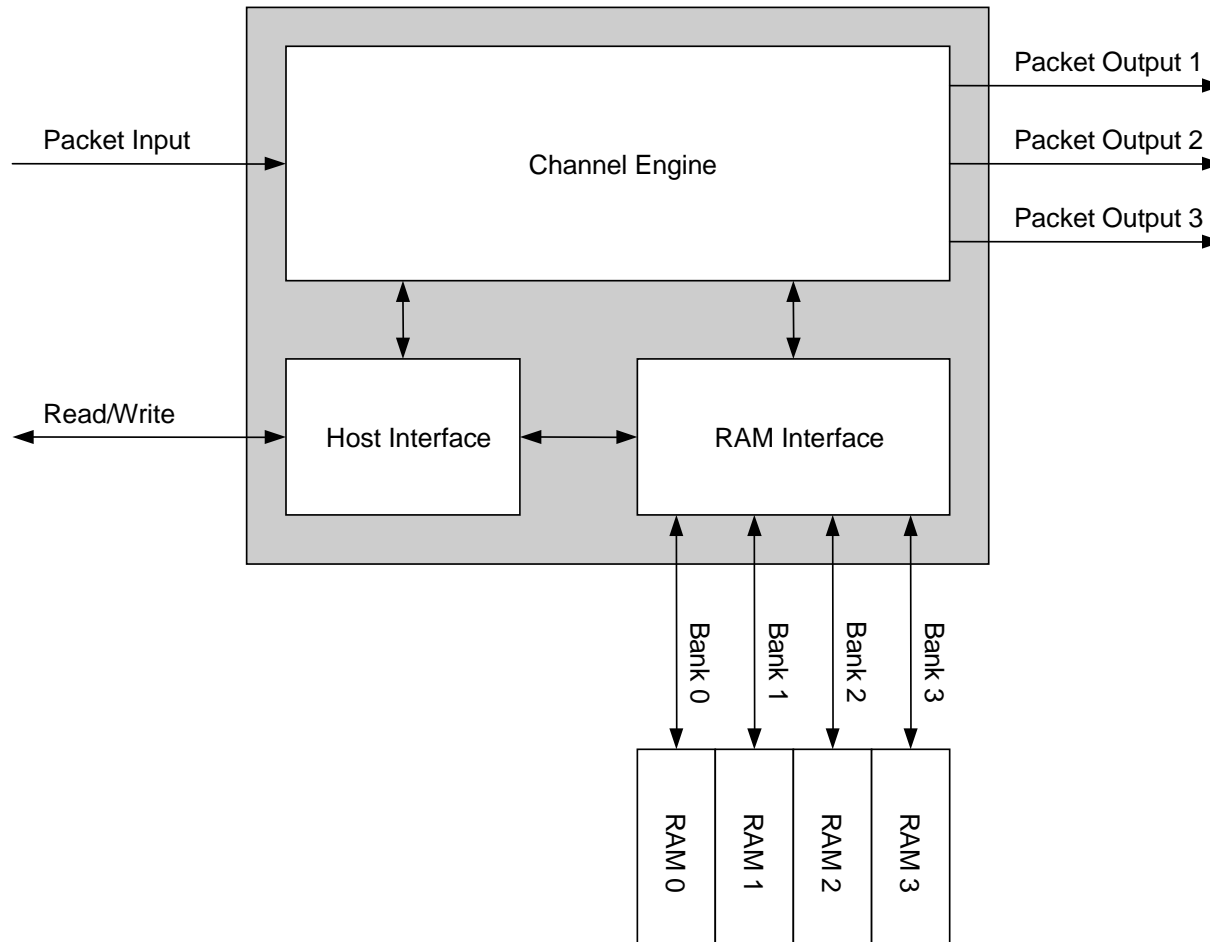


- Generate VMM Compliant
  - Directory structure
  - Drivers, monitors shells
  - VIP shells
- Generate build/run scripts
- Templates are populated by user
- Original templates customizable
- Examples
  - Standalone DUT
  - SOC

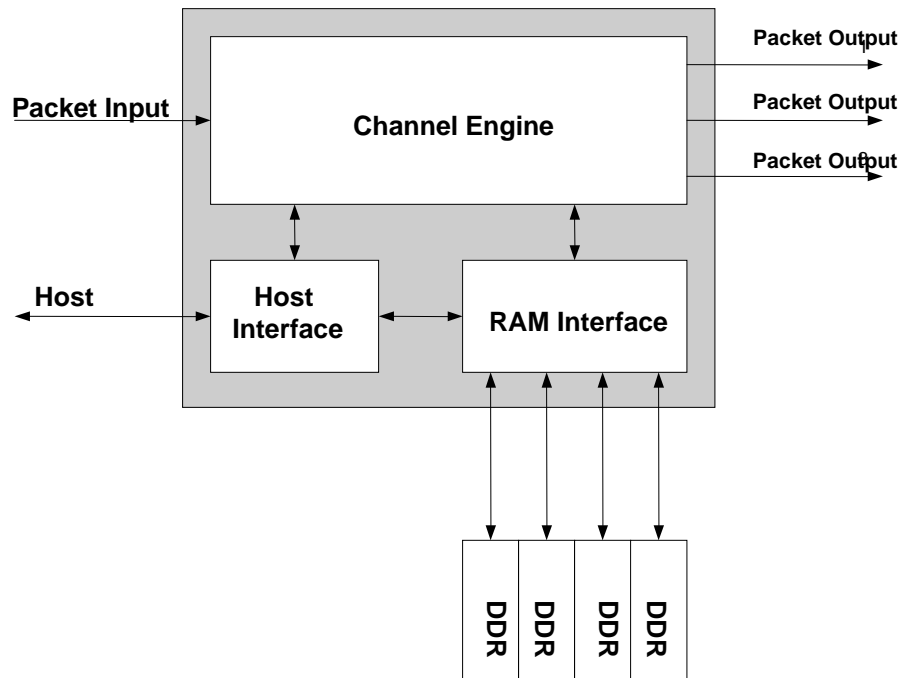
# Automatic Testbench Generation: SVF-TG



# PW Router Example (Standalone DUT)



# Automatic Testbench Generation: Router Example



## svfTG

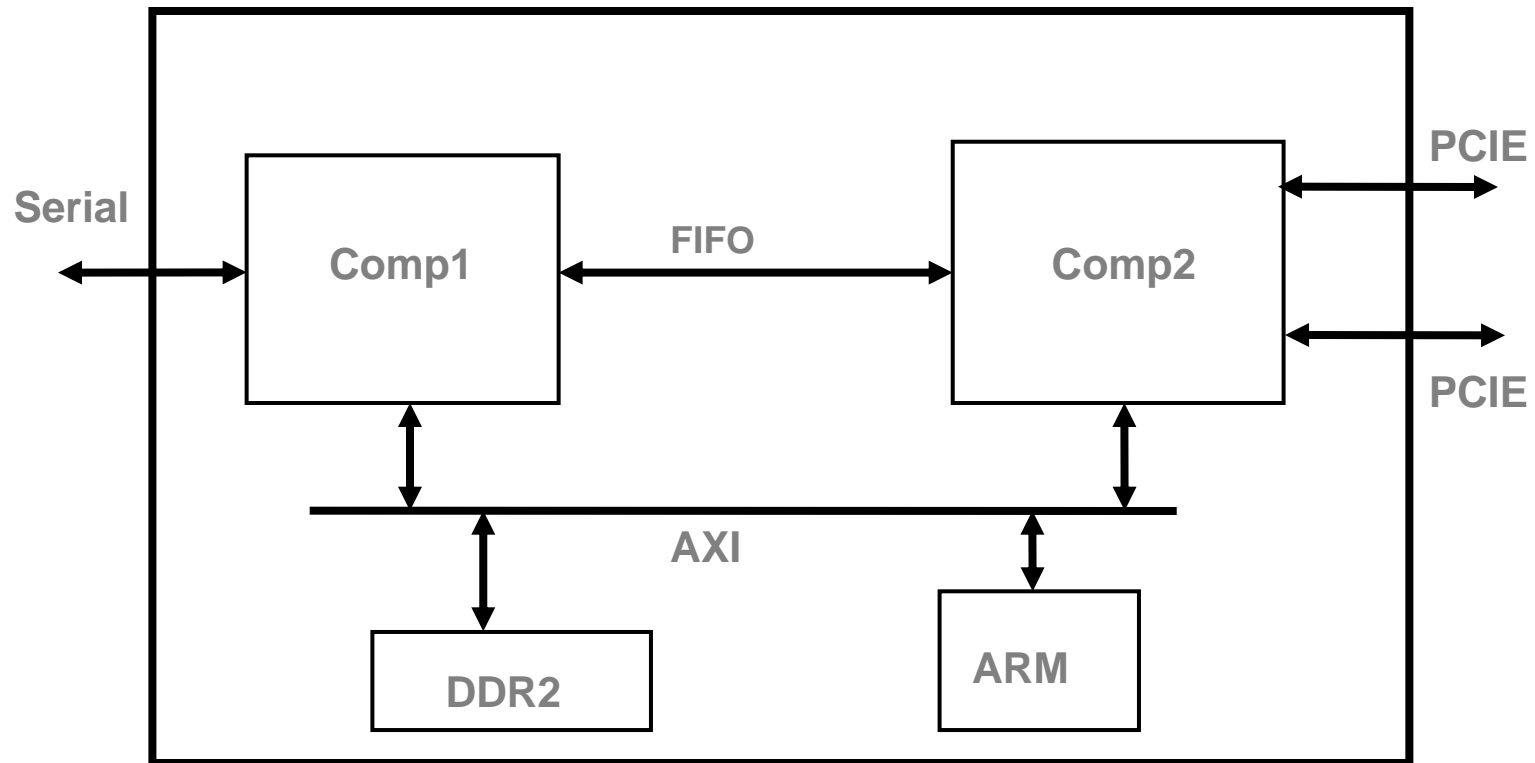
```

-p pwr
-vip "po[3]"
-vip pi
-vip host
-vip "ddr[4]"
-env sys
    
```

```

# Name of the executable
# Name of project
# 3 instances of the PO vip
# 1 instance of PI vip
# 1 instance of the host vip
# 4 instances of the DDR vip
# sys environment
    
```

# Automatic Testbench Generation: SOC Example



```

svfTG                # Name of the executable

-p snug              # Name of project

-vip "pcie[2]"        # 2 instances of the PCI-E vip

-vip ser              # 1 instance of ser vip

-vip fifo             # 1 instance of the fifo vip

-vip axi              # 1 instance of the axi interface

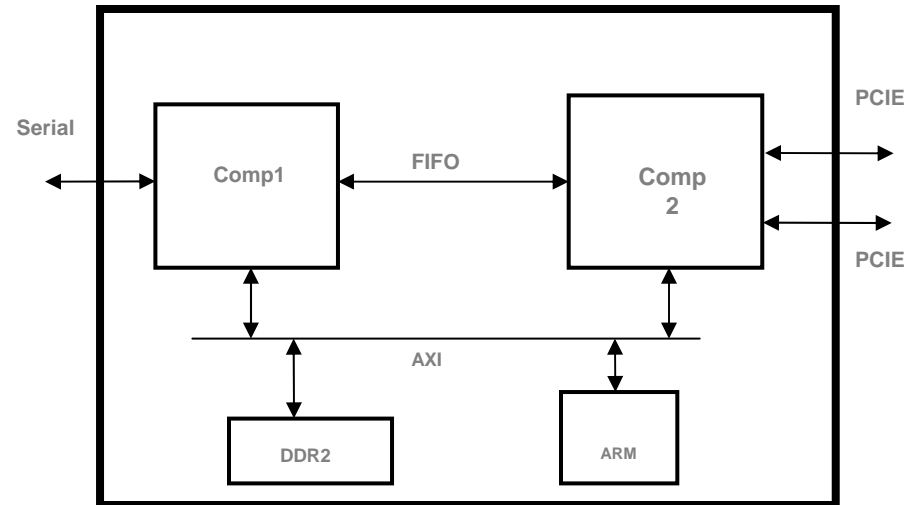
-env comp1            # comp1 environment

-te "comp1 grp1 t1 t2" # comp1 environment has
                       # tests t1, t2 in test group grp1

-env sys              # sys environment

-te "sys sys_comp1 sys_c1_test" # sys environment has sv_c1_test in test
                                # group sys_comp1

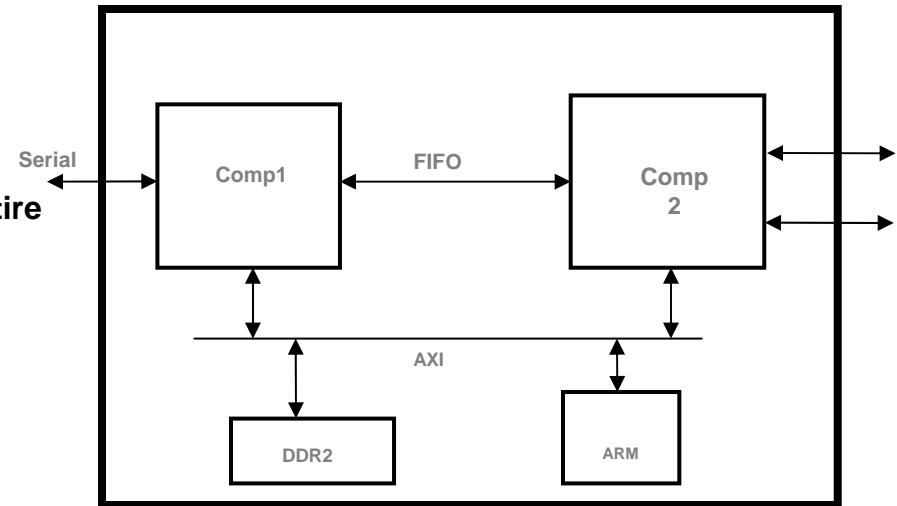
-te "sys sys sys_test" # sys environment has
                       # test sys_test in test_group sys
  
```



**DIRECTORY STRUCTURE DESCRIPTION**  
Automatically generated by SVF-TG

```
=====
```

<b>SNUG/</b>	
<b>verif/</b>	Top verification directory
<b>bin/</b>	Scripts for this project
<b>doc/</b>	Verification documentation
<b>common/</b>	Common directory used by the entire project
<b>include/</b>	Common include directory
<b>src/</b>	Common source file directory
snug_log.sv	Extended from vmm_log
snug_data.sv	Extended from vmm_data
snug_notify.sv	Extended from vmm_notify
snug_xactor.sv	Extended from vmm_xactor
snug_env.sv	Extended from vmm_env
...	
<b>src.list</b>	Build list for common dir



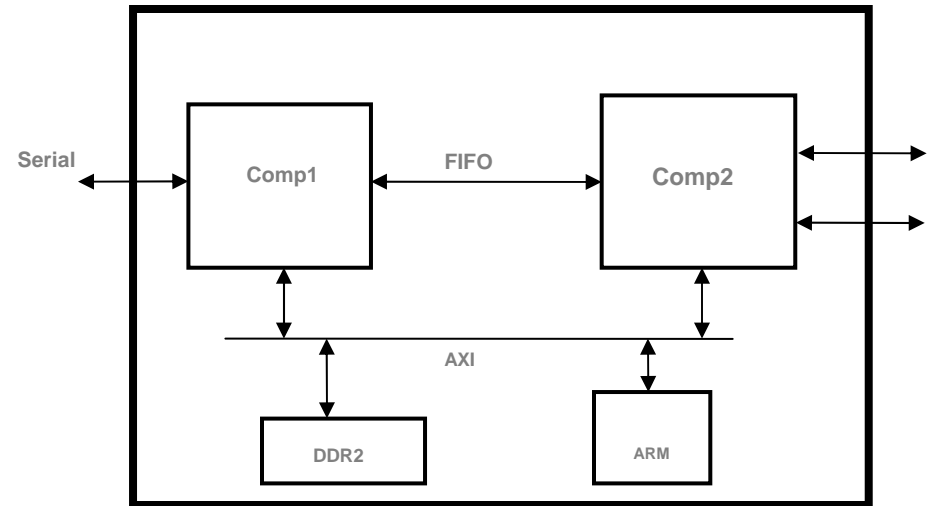
## DIRECTORY STRUCTURE DESCRIPTION

Automatically generated by SVF-TG

=====

..continued

vips/ pcie/	Directory containing all Verification IPs Distribution for VIP pcie
ser/	Distribution for VIP ser
axi/	Distribution for VIP axi
...	
comp1/ env/	Top-level directory for comp1 Verification environment files
...	
tests/ grp1/	Top level tests for comp1 Group of tests categorized under grp1
...	
sys/	Top-level directory for sys
...	



# Automatic Testbench Generation: SVF-TG

**DIRECTORY STRUCTURE DESCRIPTION**  
Automatically generated by SVF-TG

=====

<b>vips/</b>	<b>Directory containing all Verific</b>
<b>pcie/</b>	<b>Distribution for VIP pcie</b>
<b>doc/</b>	<b>Documentation for pcie VIP</b>
<b>include/</b>	<b>Include directory for pcie VIP</b>
<b>pcie_defines.sv</b>	<b>pcie specific defines</b>
<b>pcie_if.sv</b>	<b>pcie interface definitions</b>
<b>src/</b>	<b>Source files for pcie VIP</b>
<b>pcie.sv</b>	<b>Main file for pcie</b>
<b>pcie_cfg.sv</b>	<b>Configuration file for pcie</b>
<b>pcie_data.sv</b>	<b>Data class extended from snug_data</b>
<b>pcie_checker.sv</b>	<b>Checker class extended from snug_xactor</b>
<b>pcie_gen.sv</b>	<b>Generator class extended from snug_data_atomic_gen</b>
<b>pcie_drv.sv</b>	<b>Driver class extended from snug_xactor</b>
<b>pcie_mon.sv</b>	<b>Monitor class extended from snug_xactor</b>
<b>src.list</b>	<b>List of source files pcie</b>
<b>ser/</b>	<b>Distribution for VIP ser</b>

VIP specific header files

...

# Automatic Testbench Generation: SVF-TG

**DIRECTORY STRUCTURE DESCRIPTION**  
Automatically generated by SVF-TG

=====

<b>vips/</b>	<b>Directory containing all Verification IPs</b>
<b>pcie/</b>	<b>Distribution for VIP pcie</b>
<b>doc/</b>	<b>Documentation for pcie VIP</b>
<b>include/</b>	<b>Include directory for pcie VIP</b>
<b>pcie_defines.sv</b>	<b>pcie specific defines</b>
<b>pcie_if.sv</b>	<b>pcie interface definitions</b>
<b>src/</b>	<b>Source files for pcie VIP</b>
<b>pcie.sv</b>	<b>Main file for pcie</b>
<b>pcie_cfg.sv</b>	<b>Configuration file for pcie</b>
<b>pcie_data.sv</b>	<b>Data class extended from snug_data</b>
<b>pcie_checker.sv</b>	<b>Checker class extended from snug_xactor</b>
<b>pcie_gen.sv</b>	<b>Generator class extended from snug_data_atomic_gen</b>
<b>pcie_drv.sv</b>	<b>Driver class extended from snug_xactor</b>
<b>pcie_mon.sv</b>	<b>Monitor class extended from snug_xactor</b>
<b>src.list</b>	<b>List of source files pcie</b>
<b>ser/</b>	<b>Distribution for VIP ser</b>

VIP configuration

...

# Automatic Testbench Generation: SVF-TG

**DIRECTORY STRUCTURE DESCRIPTION**  
Automatically generated by SVF-TG

=====

<b>vips/</b>	<b>Directory containing all Verification IPs</b>
<b>pcie/</b>	<b>Distribution for VIP pcie</b>
<b>doc/</b>	<b>Documentation for pcie VIP</b>
<b>include/</b>	<b>Include directory for pcie VIP</b>
<b>pcie_defines.sv</b>	<b>pcie specific defines</b>
<b>pcie_if.sv</b>	<b>pcie interface definitions</b>
<b>src/</b>	<b>Source files for pcie VIP</b>
<b>pcie.sv</b>	<b>Main file for pcie</b>
<b>pcie_cfg.sv</b>	<b>Configuration file for pcie</b>
<b>pcie_data.sv</b>	<b>Data class extended from snug_data</b>
<b>pcie_checker.sv</b>	<b>Checker class extended from snug_xactor</b>
<b>pcie_gen.sv</b>	<b>Generator class extended from snug_data_atomic_gen</b>
<b>pcie_drv.sv</b>	<b>Driver class extended from snug_xactor</b>
<b>pcie_mon.sv</b>	<b>Monitor class extended from snug_xactor</b>
<b>src.list</b>	<b>List of source files pcie</b>
<b>ser/</b>	<b>Distribution for VIP ser</b>

VIP transaction class

...

# Automatic Testbench Generation: SVF-TG

**DIRECTORY STRUCTURE DESCRIPTION**  
Automatically generated by SVF-TG

=====

<b>vips/</b>	<b>Directory containing all Verification IPs</b>
<b>pcie/</b>	<b>Distribution for VIP pcie</b>
<b>doc/</b>	<b>Documentation for pcie VIP</b>
<b>include/</b>	<b>Include directory for pcie VIP</b>
<b>pcie_defines.sv</b>	<b>pcie specific defines</b>
<b>pcie_if.sv</b>	<b>pcie interface definitions</b>
<b>src/</b>	<b>Source files for pcie VIP</b>
<b>pcie.sv</b>	<b>Main file for pcie</b>
<b>pcie_cfg.sv</b>	<b>Configuration file for pcie</b>
<b>pcie_data.sv</b>	<b>Data class extended from snug_data</b>
<b>pcie_checker.sv</b>	<b>Checker class extended from snug_xactor</b>
<b>pcie_gen.sv</b>	<b>Generator class extended from snug_data_atomic_gen</b>
<b>pcie_drv.sv</b>	<b>Driver class extended from snug_xactor</b>
<b>pcie_mon.sv</b>	<b>Monitor class extended from snug_xactor</b>
<b>src.list</b>	<b>List of source files pcie</b>
<b>ser/</b>	<b>Distribution for VIP ser</b>



...

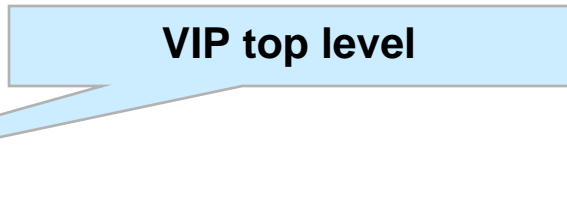
# Automatic Testbench Generation: SVF-TG

**DIRECTORY STRUCTURE DESCRIPTION**  
Automatically generated by SVF-TG

=====

<b>vips/</b>	<b>Directory containing all Verification IPs</b>
<b>pcie/</b>	<b>Distribution for VIP pcie</b>
<b>doc/</b>	<b>Documentation for pcie VIP</b>
<b>include/</b>	<b>Include directory for pcie VIP</b>
<b>pcie_defines.sv</b>	<b>pcie specific defines</b>
<b>pcie_if.sv</b>	<b>pcie interface definitions</b>
<b>src/</b>	<b>Source files for pcie VIP</b>
<b>pcie.sv</b>	<b>Main file for pcie</b>
<b>pcie_cfg.sv</b>	<b>Configuration file for pcie</b>
<b>pcie_data.sv</b>	<b>Data class extended from snug_data</b>
<b>pcie_checker.sv</b>	<b>Checker class extended from snug_xactor</b>
<b>pcie_gen.sv</b>	<b>Generator class extended from snug_data_atomic_gen</b>
<b>pcie_drv.sv</b>	<b>Driver class extended from snug_xactor</b>
<b>pcie_mon.sv</b>	<b>Monitor class extended from snug_xactor</b>
<b>src.list</b>	<b>List of source files pcie</b>
<b>ser/</b>	<b>Distribution for VIP ser</b>

VIP top level



...

# Automatic Testbench Generation: SVF-TG

**DIRECTORY STRUCTURE DESCRIPTION**  
Automatically generated by SVF-TG

=====

<b>vips/</b>	<b>Directory containing all Verification IPs</b>
<b>pcie/</b>	<b>Distribution for VIP pcie</b>
<b>doc/</b>	<b>Documentation for pcie VIP</b>
<b>include/</b>	<b>Include directory for pcie VIP</b>
<b>pcie_defines.sv</b>	<b>pcie specific defines</b>
<b>pcie_if.sv</b>	<b>pcie interface definitions</b>
<b>src/</b>	<b>Source files for pcie VIP</b>
<b>pcie.sv</b>	<b>Main file for pcie</b>
<b>pcie_cfg.sv</b>	<b>Configuration file for pcie</b>
<b>pcie_data.sv</b>	<b>Data class extended from snug_data</b>
<b>pcie_checker.sv</b>	<b>Checker class extended from snug_xactor</b>
<b>pcie_gen.sv</b>	<b>Generator class extended from snug_data_atomic_gen</b>
<b>pcie_drv.sv</b>	<b>Driver class extended from snug_xactor</b>
<b>pcie_mon.sv</b>	<b>Monitor class extended from snug_xactor</b>
<b>src.list</b>	<b>List of source files pcie</b>
<b>ser/</b>	<b>Distribution for VIP ser</b>

...



VIP build list



# Automatic Testbench Generation: SVF-TG



```

Automatically generated pcie_if.sv
//-----
// Copyright (c) 2001-2007 Paradigm Works, Inc.
// http://www.paradigm-works.com
//
// Licensed under the Apache License,
// Version 2.0 (the "License");
// you may not use this file except in compliance
// with the License. You may obtain a copy of the
// License at
// http://www.apache.org/licenses/LICENSE-2.0
// Unless required by applicable law or agreed to in
// writing, software distributed under the License is distributed
// on an "AS IS" BASIS, WITHOUT WARRANTIES OR
// CONDITIONS OF ANY KIND, either express or implied.
// See the License for the specific language governing
// permissions and limitations under the License.
//-----

```

```

/*****
* Author:      $Author:$
* File:        $File:$
* Revision:    $Revision:$
* Date:        $Date:$
*****/
* Interface file pcie_if.sv for vip pcie in snug verification environment
*****/

```

```

`ifndef SNUG_PCIE_IF
`define SNUG_PCIE_IF

interface pcie_if(input wire clk);
  logic [15:0] rdata;
  logic [15:0] wdata;
  logic write;

  // Add your own signals

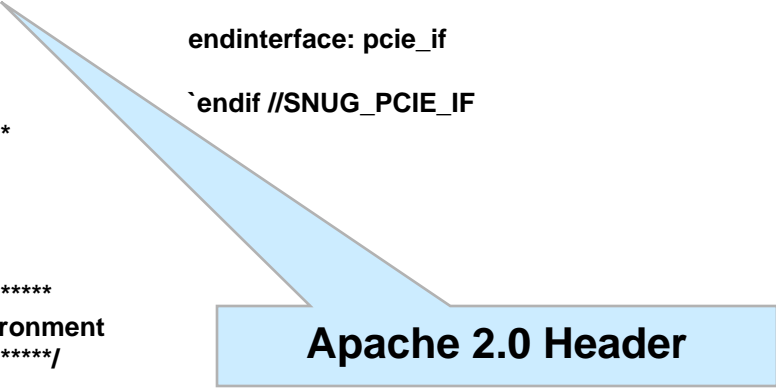
  // For the DUT
  modport DUT( input wdata, output rdata, input write);

  // For the testbench
  modport TB( input rdata, output wdata, output write);

endinterface: pcie_if

`endif //SNUG_PCIE_IF

```



Apache 2.0 Header



# Automatic Testbench Generation: SVF-TG



## Automatically generated pcie\_if.sv

```
//-----
// Copyright (c) 2001-2007 Paradigm Works, Inc.
// http://www.paradigm-works.com
//
// Licensed under the Apache License,
// Version 2.0 (the "License");
// you may not use this file except in compliance
// with the License. You may obtain a copy of the
// License at
// http://www.apache.org/licenses/LICENSE-2.0
// Unless required by applicable law or agreed to in
// writing, software distributed under the License is distributed
// on an "AS IS" BASIS, WITHOUT WARRANTIES OR
// CONDITIONS OF ANY KIND, either express or implied.
// See the License for the specific language governing
// permissions and limitations under the License.
//-----
/*****
* Author:      $Author:$
* File:        $File:$
* Revision:    $Revision:$
* Date:        $Date:$
*****
* Interface file pcie_if.sv for vip pcie in snug verification environment
*****/
```

```
`ifndef SNUG_PCIE_IF
`define SNUG_PCIE_IF

interface pcie_if(input wire clk);
  logic [15:0] rdata;
  logic [15:0] wdata;
  logic write;

  // Add your own signals

  // For the DUT
  modport DUT( input wdata, output rdata, input write);

  // For the testbench
  modport TB( input rdata, output wdata, output write);

endinterface: pcie_if

`endif //SNUG_PCIE_IF
```





# Automatic Testbench Generation: SVF-TG



## Automatically generated pcie\_if.sv

```
//-----
// Copyright (c) 2001-2007 Paradigm Works, Inc.
// http://www.paradigm-works.com
//
// Licensed under the Apache License,
// Version 2.0 (the "License");
// you may not use this file except in compliance
// with the License. You may obtain a copy of the
// License at
// http://www.apache.org/licenses/LICENSE-2.0
// Unless required by applicable law or agreed to in
// writing, software distributed under the License is distributed
// on an "AS IS" BASIS, WITHOUT WARRANTIES OR
// CONDITIONS OF ANY KIND, either express or implied.
// See the License for the specific language governing
// permissions and limitations under the License.
//-----
/*****
* Author:      $Author:$
* File:        $File:$
* Revision:    $Revision:$
* Date:        $Date:$
*****/
* Interface file pcie_if.sv for vip pcie in snug verification
* environment
*****/
```

```
`ifndef SNUG_PCIE_IF
`define SNUG_PCIE_IF

interface pcie_if(input wire clk);
  logic [15:0] rdata;
  logic [15:0] wdata;
  logic write;

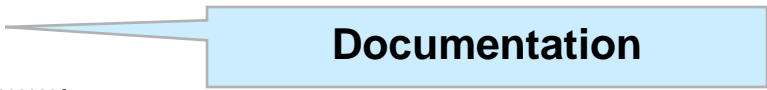
  // Add your own signals

  // For the DUT
  modport DUT( input wdata, output rdata, input write);

  // For the testbench
  modport TB( input rdata, output wdata, output write);

endinterface: pcie_if

`endif //SNUG_PCIE_IF
```



## Automatically generated pcie\_if.sv

```
//-----
// Copyright (c) 2001-2007 Paradigm Works, Inc.
// http://www.paradigm-works.com
//
// Licensed under the Apache License,
// Version 2.0 (the "License");
// you may not use this file except in compliance
// with the License. You may obtain a copy of the
// License at
// http://www.apache.org/licenses/LICENSE-2.0
// Unless required by applicable law or agreed to in
// writing, software distributed under the License is distributed
// on an "AS IS" BASIS, WITHOUT WARRANTIES OR
// CONDITIONS OF ANY KIND, either express or implied.
// See the License for the specific language governing
// permissions and limitations under the License.
//-----
/*****
* Author:      $Author:$
* File:        $File:$
* Revision:    $Revision:$
* Date:        $Date:$
*****/
* Interface file pcie_if.sv for vip pcie in snug verification environment
*****/
```

```
`ifndef SNUG_PCIE_IF
`define SNUG_PCIE_IF

interface pcie_if(input wire clk);
    logic [15:0] rdata;
    logic [15:0] wdata;
    logic write;

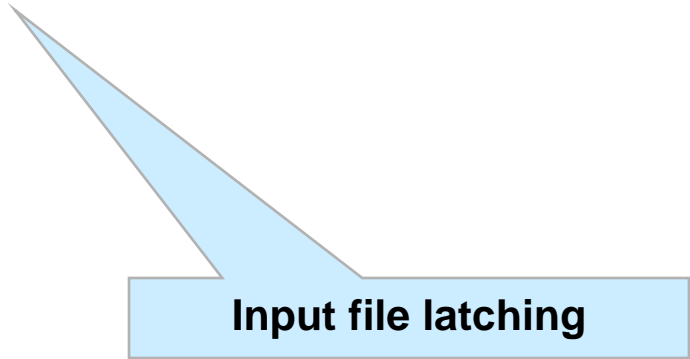
    // Add your own signals

    // For the DUT
    modport DUT( input wdata, output rdata, input write);

    // For the testbench
    modport TB( input rdata, output wdata, output write);

endinterface: pcie_if

`endif //SNUG_PCIE_IF
```



## .... Automatically generated pcie\_drv.sv

```

/*****
* Top level driver file pcie_drv.sv for snug verification environment
*****/
`include "vmm.sv"
typedef class pcie_drv;
class pcie_drv_callbacks extends vmm_xactor_callbacks;

    virtual task pre_drv(pcie_drv drv, pcie_data tr);
    endtask // pre_drv

    virtual task post_drv(pcie_drv drv, pcie_data tr);
    endtask // post_drv

endclass

class pcie_drv extends snug_xactor:
    pcie_cfg cfg;
    pcie_data_channel in_chan;

    function new( string instance,
                int stream_id=-1,
                pcie_cfg cfg=null,
                pcie_data_channel in_chan=null);
        ...deleted for clarity....

    endfunction // new

```

```

protected virtual task main();
begin
    // Common Code
    fork
        super.main();
    join_none
    // Add your own

    // Commom code
    while(1) begin
        super.wait_if_stopped();
        transmit_t();
    end
endtask // main

task transmit_t();
pcie_data tr=new;
begin
    if (in_chan.level()==0) begin
        notify.indicate(XACTOR_IDLE); notify.reset(XACTOR_BUSY);
    end
    in_chan.get(tr);
    notify.indicate(XACTOR_BUSY); notify.reset(XACTOR_IDLE);
    `vmm_callback(pcie_drv_callbacks,pre_drv(this,tr));

    // Add your own, Do the actual driving

    `vmm_callback(pcie_drv_callbacks,post_drv(this,tr));
end
endtask // transmit_t

```

**VIP Driver**

# Automatic Testbench Generation: SVF-TG

## ... Automatically generated pcie\_drv.sv

```

/*****
* Top level driver file pcie_drv.sv for snug verification environment
*****/
`include "vmm.sv"
typedef class pcie_drv;
class pcie_drv_callbacks extends vmm_xactor_callbacks;

    virtual task pre_drv(pcie_drv drv, pcie_data tr);
    endtask // pre_drv

    virtual task post_drv(pcie_drv drv, pcie_data tr);
    endtask // post_drv

endclass

class pcie_drv extends snug_xactor;
    pcie_cfg cfg;
    pcie_data_channel in_chan;

    function new( string instance,
                  int stream_id=-1,
                  pcie_cfg cfg=null,
                  pcie_data_channel in_chan=null);
        ...deleted for clarity....
    endfunction // new

```



**VIP main**

```

protected virtual task main();
begin
    // Common Code
    fork
        super.main();
    join_none
    // Add your own

    // Commom code
    while(1) begin
        super.wait_if_stopped();
        transmit_t();
    end
end
endtask // main

task transmit_t();
    pcie_data tr=new;
    begin
        if (in_chan.level()==0) begin
            notify.indicate(XACTOR_IDLE); notify.reset(XACTOR_BUSY);
        end
        in_chan.get(tr);
        notify.indicate(XACTOR_BUSY); notify.reset(XACTOR_IDLE);
        `vmm_callback(pcie_drv_callbacks,pre_drv(this,tr));

        // Add your own, Do the actual driving

        `vmm_callback(pcie_drv_callbacks,post_drv(this,tr));
    end
endtask // transmit_t

```

# Automatic Testbench Generation: SVF-TG



## ... Automatically generated pcie\_drv.sv

```

/*****
* Top level driver file pcie_drv.sv for snug verification environment
*****/
`include "vmm.sv"
typedef class pcie_drv;
class pcie_drv_callbacks extends vmm_xactor_callbacks;

    virtual task pre_drv(pcie_drv drv, pcie_data tr);
    endtask // pre_drv

    virtual task post_drv(pcie_drv drv, pcie_data tr);
    endtask // post_drv

endclass

class pcie_drv extends snug_xactor;
    pcie_cfg cfg;
    pcie_data_channel in_chan;

    function new( string instance,
                 int stream_id=-1,
                 pcie_cfg cfg=null,
                 pcie_data_channel in_chan=null);
        ...deleted for clarity....
    endfunction // new

```

```

protected virtual task main();
begin
    // Common Code
    fork
        super.main();
    join_none
    // Add your own

    // Commom code
    while(1) begin
        super.wait_if_stopped();
        transmit_t();
    end
end
endtask // main

task transmit_t();
    pcie_data tr=new;
    begin
        if (in_chan.level()==0) begin
            notify.indicate(XACTOR_IDLE); notify.reset(XACTOR_BUSY);
        end
        in_chan.get(tr);
        notify.indicate(XACTOR_BUSY); notify.reset(XACTOR_IDLE);
        `vmm_callback(pcie_drv_callbacks,pre_drv(this,tr));

        // Add your own, Do the actual driving

        `vmm_callback(pcie_drv_callbacks,post_drv(this,tr));
    end
endtask // transmit_t

```

VIP callbacks

# Automatic Testbench Generation: SVF-TG



## ... Automatically generated pcie\_drv.sv

```

/*****
 * Top level driver file pcie_drv.sv for snug verification environment
 *****/
`include "vmm.sv"
typedef class pcie_drv;
class pcie_drv_callbacks extends vmm_xactor_callbacks;

    virtual task pre_drv(pcie_drv drv, pcie_data tr);
    endtask // pre_drv

    virtual task post_drv(pcie_drv drv, pcie_data tr);
    endtask // post_drv

endclass

class pcie_drv extends snug_xactor;
    pcie_cfg cfg;
    pcie_data_channel in_chan;

    function new( string instance,
                int stream_id=-1,
                pcie_cfg cfg=null,
                pcie_data_channel in_chan=null);
        ...deleted for clarity....

    endfunction // new

```

```

protected virtual task main();
begin
    // Common Code
    fork
        super.main();
    join_none
    // Add your own

    // Commom code
    while(1) begin
        super.wait_if_stopped();
        transmit_t();
    end
end
endtask // main

task transmit_t();
    pcie_data tr=new;
    begin
        if (in_chan.level()==0) begin
            notify.indicate(XACTOR_IDLE); notify.reset(XACTOR_BUSY);
        end
        in_chan.get(tr);
        notify.indicate(XACTOR_BUSY); notify.reset(XACTOR_IDLE);
        `vmm_callback(pcie_drv_callbacks,pre_drv(this,tr));

        // Add your own, Do the actual driving

        `vmm_callback(pcie_drv_callbacks,post_drv(this,tr));
    end
endtask // transmit_t

```

**User specific code**

## Automatically generated sys\_env.sv

```

....
`include "pcie_if.sv"
`include "ser_if.sv"
`include "fifo_if.sv"
`include "axi_if.sv"
...
class SNUG_sys_env extends snug_
...
static vmm_log
sys_env_cfg
sys_reg_map
snug_scoreboard

log = new ("S...");
cfg;
reg_ma
sys_sb;

pcie_inst[2];
pcie_port[2];
ser_ins;
ser_port;
fifo_inst;
fifo_port;
axi_inst;
axi_port;

// Various vips
pcie_xtr
virtual pcie_if.TB
ser_xtr
virtual ser_if.TB
fifo_xtr
virtual fifo_if.TB
axi_xtr
virtual axi_if.TB

....
// Build the xactors
cfg.pcie_cfg[1].pcie_port = this.pcie_port[1];
pcie_inst[1] = new( "sys_pcie_xactor 1", "SYS_PCIE_1", 0,
                    cfg.pcie_cfg[1]);

cfg.pcie_cfg[0].pcie_port = this.pcie_port[0];
pcie_inst[0] = new( "sys_pcie_xactor 0", "SYS_PCIE_0", 0,
                    cfg.pcie_cfg[0]);

cfg.ser_cfg.ser_port = this.ser_port;
ser_inst = new( "sys_ser_xactor", "SYS_SER", 0, cfg.ser_cfg);

// Start up the rest of the transactors
fork
begin
`vmm_debug(this.log,"Started transactor pcie[0]...");
pcie_inst[0].start_xactor();
`vmm_debug(this.log,"Started transactor pcie[1]...");
pcie_inst[1].start_xactor();
`vmm_debug(this.log,"Started transactor ser...");
ser_inst.start_xactor();
.....
end
join
.....

```

**Include appropriate files**

## Automatically generated sys\_env.sv

```

....
`include "pcie_if.sv"
`include "ser_if.sv"
`include "fifo_if.sv"
`include "axi_if.sv"
...
class SNUG_sys_env extends snug_env;
...
    static vmm_log
    sys_env_cfg
    sys_reg_map
    snug_scoreboard

    // Various vips
    pcie_xtr
    virtual pcie_if.TB
    ser_xtr
    virtual ser_if.TB
    fifo_xtr
    virtual fifo_if.TB
    axi_xtr
    virtual axi_if.TB

    log = new ("SNUG","ENV");
    cfg;
    reg_ma
    sys_sb;

    pcie_inst[2];
    pcie_port[2];
    ser_ins;
    ser_port;
    fifo_inst;
    fifo_port;
    axi_inst;
    axi_port;

    // Build the xactors
    cfg.pcie_cfg[1].pcie_port = this.pcie_port[1];
    pcie_inst[1] = new( "sys_pcie_xactor 1", "SYS_PCIE_1", 0,
        cfg.pcie_cfg[1]);

    cfg.pcie_cfg[0].pcie_port = this.pcie_port[0];
    pcie_inst[0] = new( "sys_pcie_xactor 0", "SYS_PCIE_0", 0,
        cfg.pcie_cfg[0]);

    cfg.ser_cfg.ser_port = this.ser_port;
    ser_inst = new( "sys_ser_xactor", "SYS_SER", 0, cfg.ser_cfg);

    // Start up the rest of the transactors
    fork
    begin
        `vmm_debug(this.log,"Started transactor pcie[0]...");
        pcie_inst[0].start_xactor();
        `vmm_debug(this.log,"Started transactor pcie[1]...");
        pcie_inst[1].start_xactor();
        `vmm_debug(this.log,"Started transactor ser...");
        ser_inst.start_xactor();
    ....
    end
    join
    ....

```

**Declare components**

## Automatically generated sys\_env.sv

```

....
`include "pcie_if.sv"
`include "ser_if.sv"
`include "fifo_if.sv"
`include "axi_if.sv"
...
class SNUG_sys_env extends snug_env;
...
    static vmm_log
    sys_env_cfg
    sys_reg_map
    snug_scoreboard

    log = new ("SNUG","ENV");
    cfg;
    reg_m
    sys_sb;

    // Various vips
    pcie_xtr
    virtual pcie_if.TB
    ser_xtr
    virtual ser_if.TB
    fifo_xtr
    virtual fifo_if.TB
    axi_xtr
    virtual axi_if.TB

    pcie_inst[2];
    pcie_port[2];
    ser_ins;
    ser_port;
    fifo_inst;
    fifo_port;
    axi_inst;
    axi_port;

    // Build the xactors
    cfg.pcie_cfg[1].pcie_port = this.pcie_port[1];
    pcie_inst[1] = new( "sys_pcie_xactor 1", "SYS_PCIE_1", 0,
        cfg.pcie_cfg[1]);

    cfg.pcie_cfg[0].pcie_port = this.pcie_port[0];
    pcie_inst[0] = new( "sys_pcie_xactor 0", "SYS_PCIE_0", 0,
        cfg.pcie_cfg[0]);

    cfg.ser_cfg.ser_port = this.ser_port;
    ser_inst = new( "sys_ser_xactor", "SYS_SER", 0, cfg.ser_cfg);

    // Start up the rest of the transactors
    fork
    begin
        `vmm_debug(this.log,"Started transactor pcie[0]...");
        pcie_inst[0].start_xactor();
        `vmm_debug(this.log,"Started transactor pcie[1]...");
        pcie_inst[1].start_xactor();
        `vmm_debug(this.log,"Started transactor ser...");
        ser_inst.start_xactor();
    ....
    end
    join
    ....

```

**Create VIP instances**

## Automatically generated sys\_env.sv

```

....
`include "pcie_if.sv"
`include "ser_if.sv"
`include "fifo_if.sv"
`include "axi_if.sv"
...
class SNUG_sys_env extends snug_env;
...
    static vmm_log
    sys_env_cfg
    sys_reg_map
    snug_scoreboard

    log = new ("SNUG","ENV");
    cfg;
    reg_ma
    sys_sb;

    // Various vips
    pcie_xtr
    virtual pcie_if.TB
    ser_xtr
    virtual ser_if.TB
    fifo_xtr
    virtual fifo_if.TB
    axi_xtr
    virtual axi_if.TB

    pcie_inst[2];
    pcie_port[2];
    ser_inst;
    ser_port;
    fifo_inst;
    fifo_port;
    axi_inst;
    axi_port;

    // Build the xactors
    cfg.pcie_cfg[1].pcie_port = this.pcie_port[1];
    pcie_inst[1] = new( "sys_pcie_xactor 1", "SYS_PCIE_1", 0,
        cfg.pcie_cfg[1]);

    cfg.pcie_cfg[0].pcie_port = this.pcie_port[0];
    pcie_inst[0] = new( "sys_pcie_xactor 0", "SYS_PCIE_0", 0,
        cfg.pcie_cfg[0]);

    cfg.ser_cfg.ser_port = this.ser_port;
    ser_inst = new( "sys_ser_xactor", "SYS_SER", 0, cfg.ser_cfg);

    // Start up the rest of the transactors
    fork
    begin
        `vmm_debug(this.log,"Started transactor pcie[0]...");
        pcie_inst[0].start_xactor();
        `vmm_debug(this.log,"Started transactor pcie[1]...");
        pcie_inst[1].start_xactor();
        `vmm_debug(this.log,"Started transactor ser...");
        ser_inst.start_xactor();
    ....
    end
    join
    ....

```



## Automatically generated sys\_env.sv

```

....
`include "pcie_if.sv"
`include "ser_if.sv"
`include "fifo_if.sv"
`include "axi_if.sv"
...
class SNUG_sys_env extends snug_env;
...
    static vmm_log
    sys_env_cfg
    sys_reg_map
    snug_scoreboard

    // Various vips
    pcie_xtr
    virtual pcie_if.TB
    ser_xtr
    virtual ser_if.TB
    fifo_xtr
    virtual fifo_if.TB
    axi_xtr
    virtual axi_if.TB

    log = new ("SNUG","ENV");
    cfg;
    reg_ma
    sys_sb;

    pcie_inst[2];
    pcie_port[2];
    ser_ins;
    ser_port;
    fifo_inst;
    fifo_port;
    axi_inst;
    axi_port;

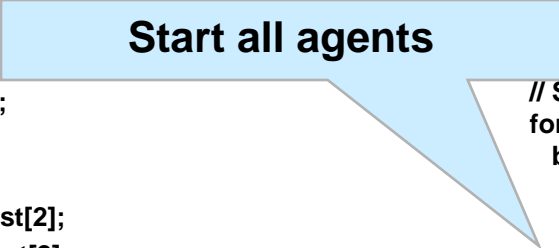
    // Build the xactors
    cfg.pcie_cfg[1].pcie_port = this.pcie_port[1];
    pcie_inst[1] = new( "sys_pcie_xactor 1", "SYS_PCIE_1", 0,
        cfg.pcie_cfg[1]);

    cfg.pcie_cfg[0].pcie_port = this.pcie_port[0];
    pcie_inst[0] = new( "sys_pcie_xactor 0", "SYS_PCIE_0", 0,
        cfg.pcie_cfg[0]);

    cfg.ser_cfg.ser_port = this.ser_port;
    ser_inst = new( "sys_ser_xactor", "SYS_SER", 0, cfg.ser_cfg);

    // Start up the rest of the transactors
    fork
    begin
        `vmm_debug(this.log,"Started transactor pcie[0]...");
        pcie_inst[0].start_xactor();
        `vmm_debug(this.log,"Started transactor pcie[1]...");
        pcie_inst[1].start_xactor();
        `vmm_debug(this.log,"Started transactor ser...");
        ser_inst.start_xactor();
    ....
    end
    join
    ....

```



## Automatically generated sys\_env.sv

```

....
`include "pcie_if.sv"
`include "ser_if.sv"
`include "fifo_if.sv"
`include "axi_if.sv"
...
class SNUG_sys_env extends snug_env;
...
    static vmm_log
    sys_env_cfg
    sys_reg_map
    snug_scoreboard

    // Various vips
    pcie_xtr
    virtual pcie_if.TB
    ser_xtr
    virtual ser_if.TB
    fifo_xtr
    virtual fifo_if.TB
    axi_xtr
    virtual axi_if.TB

    log = new ("SNUG","ENV");
    cfg;
    reg_ma
    sys_sb;

    pcie_inst[2];
    pcie_port[2];
    ser_ins;
    ser_port;
    fifo_inst;
    fifo_port;
    axi_inst;
    axi_port;

    ....
    // Build the xactors
    cfg.pcie_cfg[1].pcie_port = this.pcie_port[1];
    pcie_inst[1] = new( "sys_pcie_xactor 1", "SYS_PCIE_1", 0,
        cfg.pcie_cfg[1]);

    cfg.pcie_cfg[0].pcie_port = this.pcie_port[0];
    pcie_inst[0] = new( "sys_pcie_xactor 0", "SYS_PCIE_0", 0,
        cfg.pcie_cfg[0]);

    cfg.ser_cfg.ser_port = this.ser_port;
    ser_inst = new( "sys_ser_xactor", "SYS_SER", 0, cfg.ser_cfg);

    // Start up the rest of the transactors
    fork
    begin
        `vmm_debug(this.log,"Started transactor pcie[0]...");
        pcie_inst[0].start_xactor();
        `vmm_debug(this.log,"Started transactor pcie[1]...");
        pcie_inst[1].start_xactor();
        `vmm_debug(this.log,"Started transactor ser...");
        ser_inst.start_xactor();
    ....
    end
    join
    ....

```

**Specific comments**

- Creating and maintaining verification environments across multiple sites and groups is challenging.
- The SVF-TG tool allows teams to capture the project specific guidelines in an executable form so that complete and detailed verification environment shells can be customized and generated from scratch.
- VMM methodology helps significantly in maintaining verification environments



## Contact for SVF-TG

34

Ambar Sarkar



SystemVerilog FrameWorks™ Template Generator is available freely for generating customized VMM environments.

Please contact

http: [svf-tg.paradigm-works.com](http://svf-tg.paradigm-works.com)

email: [svf-tg@paradigm-works.com](mailto:svf-tg@paradigm-works.com)



contact

 search

home

log in join

you are here: home

links

- Paradigm-Works Home
- SVF Template Generator

log in

**Login Name**

**Password**

[Forgot your password?](#)

[New user?](#)

## SystemVerilog Frameworks Template Generator (SVF-TG)

by svf-tg — last modified 2007-04-03 02:33

SystemVerilog Frameworks<sup>(TM)</sup> Template Generator (SVF-TG) is a tool for generating a detailed boilerplate for a VMM based verification environment from scratch based on user input. The generated code is compilable with VCS version 2006.06 and later.

Do not hesitate to contact [svf-tg@paradigm-works.com](mailto:svf-tg@paradigm-works.com) if you have any comments and/or suggestions.

Click on the link below:

[SVF Template Generator](#)

Copyright 2007, Paradigm-Works, Inc.



This site conforms to the following standards:

SECTION 508 W3C AA W3C XHTML W3C CSS ANY BROWSER