



PARADIGM[®]
WORKS

**Generating
VMM-Compliant
Verification Environments**

Ning Guo



Overview

- ▶ Challenges of creating/maintaining VMM environments
- ▶ SystemVerilog FrameWorks™ Template Generator (SVF-TG)
 - ▶ SVF-TG Flow
 - ▶ Generated VMM compliant Testbench
- ▶ Customize Templates
 - ▶ Modify Directory Structure
 - ▶ Modify templates
 - ▶ Upgrade testbench
- ▶ 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

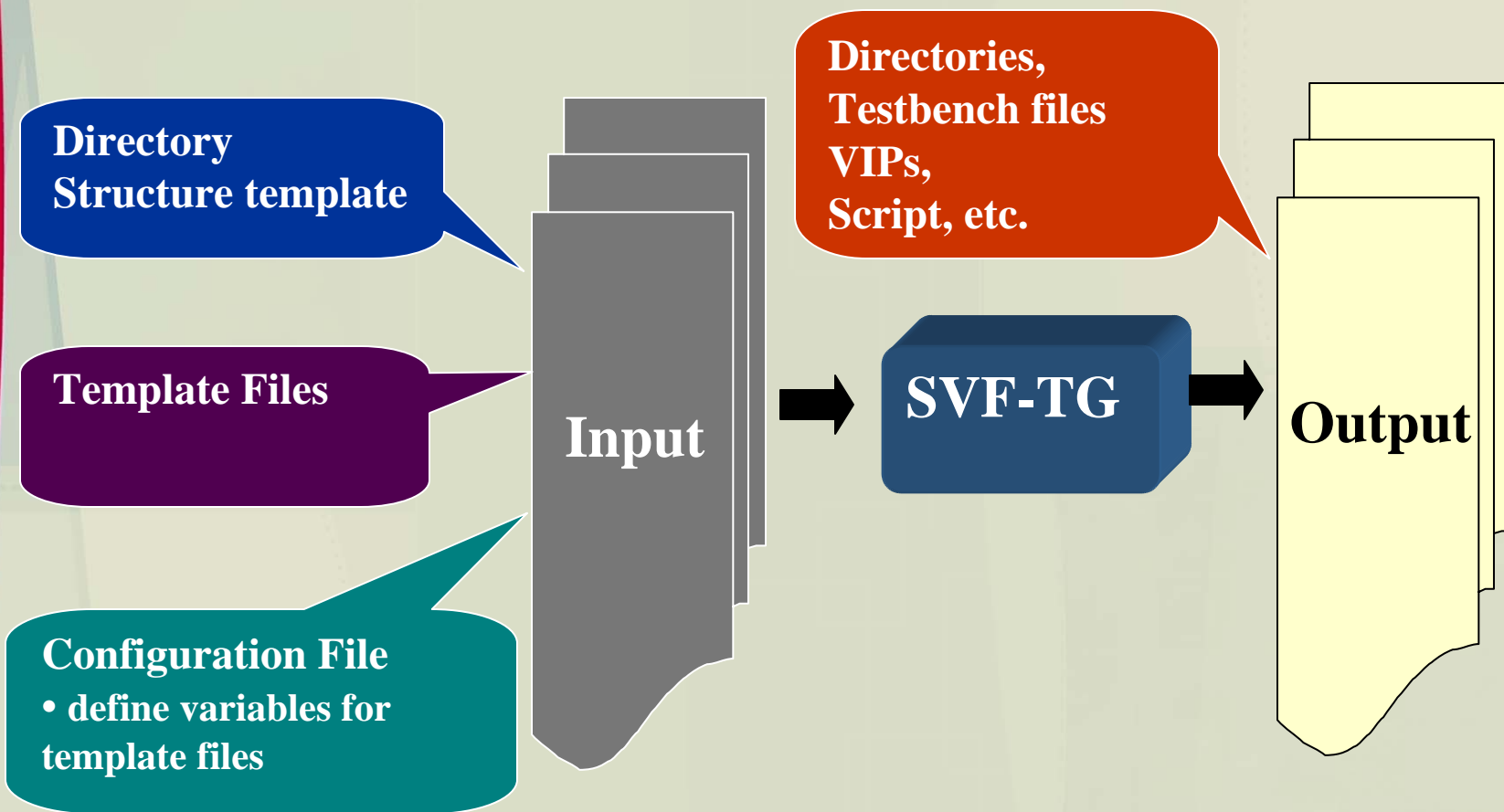


SystemVerilog FrameWorks™ Template Generator (SVF-TG)

- ▶ A tool to jumpstart a SystemVerilog testbench
 - ▶ 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

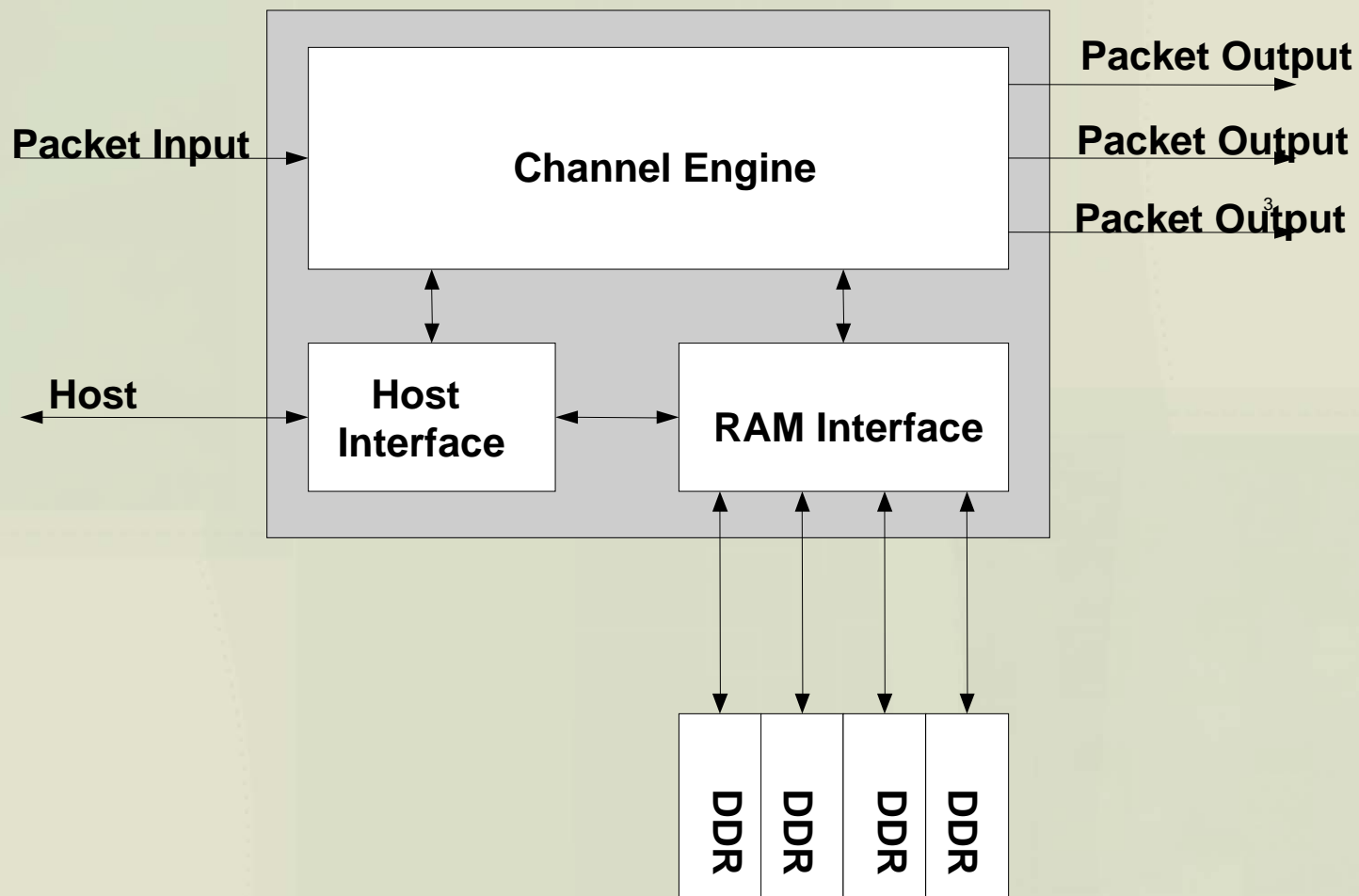


SVF-TG Flow





Generate VMM Testbench: Router Example





Router Example (con't)

```
> svfTG -p pwr -vip pi -vip host -vip "ddr[4]" -env  
sys -test "basic" -i ~/tmp  
-tt $SVF_HOME/templates/vmm_dir_struct
```

svfTG	# Name of the executable
-p pwr	# Name of project
-vip pi	# 1 instance of PI vip
-vip host	# 1 instance of the host vip
-vip "ddr[4]"	# 4 instances of the DDR vip
-env sys	# sys environment
-test "basic"	# test group
-tt \$SVF_HOME/templates/vmm_dir_struct	# Template Dir

Predefined VMM
templates



Router Example (con't)

GENERATED DIRECTORY STRUCTURE

PWR/	Project Directory
verif/	Verification database
vips/	Verification IPs
host/	Host Interface VIP
pi/	Packet Input Interface VIP
ddr/	DDR Interface VIP
sys/	Testbench and testcase area
env/	PWR Testbench
test/	Testcase area
basic/	Basic test group
 pw_README.txt	 Directory structure Doc



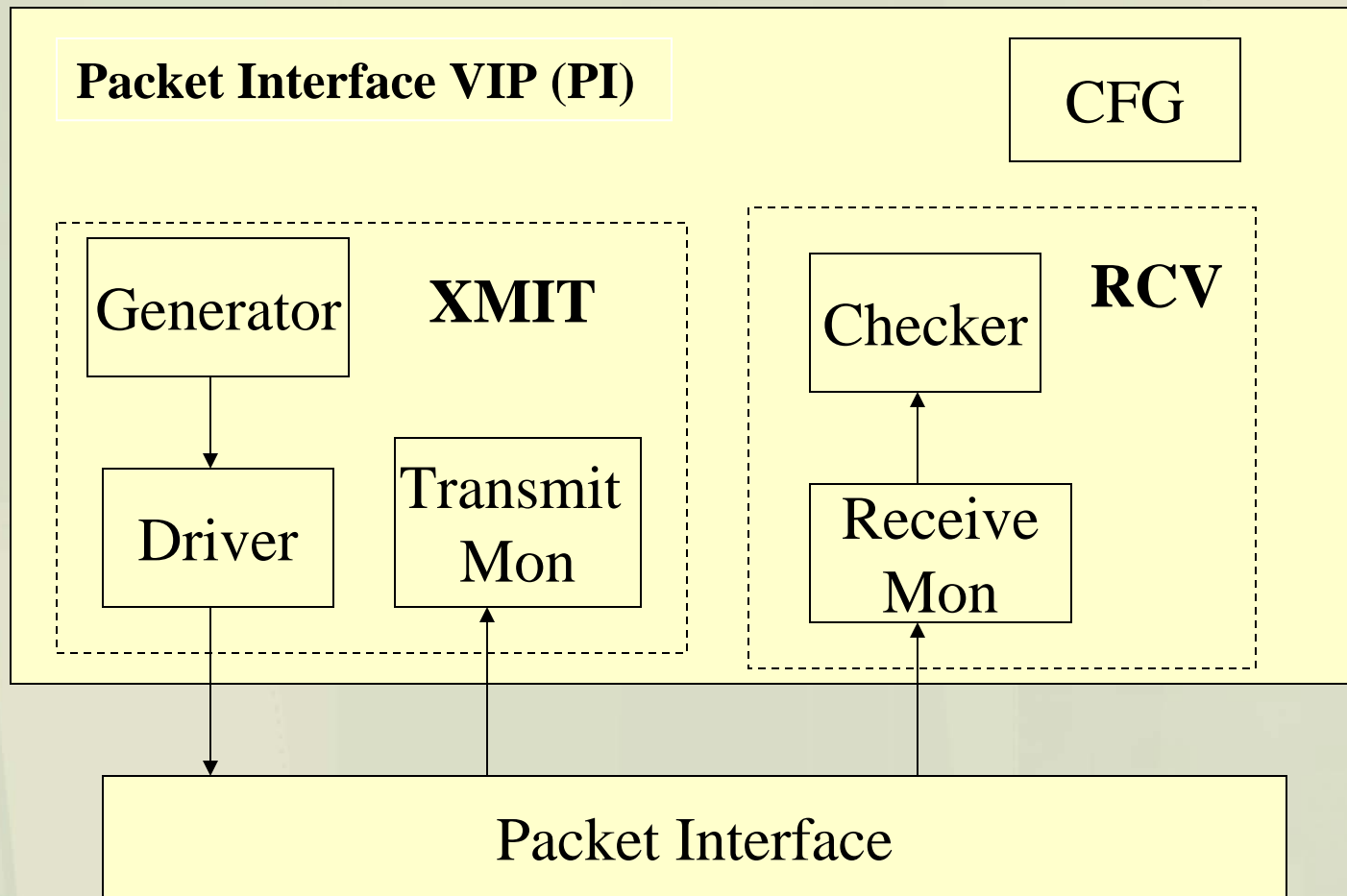
SVF-TG Generated VMM Compliant VIP

Packet Interface VIP (using predefined templates)

```
pi/include          # Directory for VIP specific includes
  pi_if.sv          Packet Interface
  pi_defines.sv     Constants,enums for packet VIP
pi/src              # VIP Components extend from vmm_xactor
  pi.sv             Top level packet Interface
  pi_cfg.sv         Configuration descriptor of Packet VIP
  pi_xmit.sv        Transmit side wrapper of Packet VIP
  pi_rcv.sv         Receive side wrapper of Packet VIP
  pi_gen.sv         Generator component, instantiated in xmit
  pi_drv.sv         Driver component, instantiated in xmit
  pi_mon.sv         Monitor component, instantiate in xmit/rcv
  pi_checker.sv     Checker component, instantiate in rcv
  pi_packet.sv     # Data structure. Extends from vmm_data
```



SVF-TG Generated VMM Compliant VIP





SVF-TG Generated pi_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 pi_if.sv for vip pi in sys environment
*****/
```

Apache 2.0 Header

Source Control

Documentation

```
`ifndef PI_IF
`define PI_IF

interface pi_if(input wire clk);
// Add your own declarations

clocking cb @(posedge clk);
    default input #1 output #1;
    // Add your own signals
endclocking: cb

// For the DUT
modport DUT( // Add your own signals
);

// For the testbench
modport TB (clocking cb);

`endif //PI_IF
```

Input file latching

Clocking block

modport



SVF-TG Generated pi_drv.sv

```
/*  
 * Top level driver file pi_drv.sv for sys environment  
 */  
`include "vmm.sv"  
typedef class pi_drv;  
class pi_drv_callbacks extends vmm_xactor_callbacks;  
    virtual task pre_drv(pi_drv drv, pi_data tr);  
endtask // pre_drv  
  
    virtual task post_drv(pi_drv drv, pi_data tr);  
endtask // post_drv  
endclass  
  
class pi_drv extends vmm_xactor;  
    pi_cfg cfg;  
    pi_data_channel in_chan;  
  
    function new(string inst,  
                 int stream_id=-1,  
                 pi_cfg cfg=null,  
                 pi_data_channel in_chan=null);  
        ...deleted for clarity....  
endfunction // new
```

VIP Callback Class

VIP Driver

User specific code

```
protected virtual task main();  
    fork  
        super.main();  
    join_none  
  
    while(1) begin  
        super.wait_if_stopped();  
        transmit_t();  
    end  
endtask // main  
  
task transmit_t();  
    pi_data tr=new;  
  
    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(pi_drv_callbacks,pre_drv(this,tr));  
    // Add your own, Do the actual driving  
  
    `vmm_callback(pi_drv_callbacks,post_drv(this,tr));  
end  
endtask // transmit_t
```

VIP main

VIP callbacks



SVF-TG Generated VMM Testbench

VMM Compliant Testbench (using predefined templates)

=====

env/

pwr_top.sv	Top level testbench module
pwr_env.sv	Top level testbench. Extends vmm_env
pwr_env_cfg.sv	Testbench configuration descriptor
pwr_env_scoreboard.sv	Scoreboard
pwr_regmap.sv	Place holder for register shadows

tests/basic

basic test group

basic_test.sv	Sample test
Makefile	Compile and run command



SVF-TG Generated sys_env.sv

Create VIP instances

```
`include "pi_if.sv"  
`include "host_if.sv"  
`include "ddr_if.sv"  
...
```

Include appropriate files

```
class PWR_sys_env extends vmm_env;  
...  
static vmm_log log = new ("SYS","ENV");
```

Declare components

```
sys_env_cfg    cfg;  
sys_reg_map    reg_map;  
pwr_scoreboard sys_sb;
```

Debug message

```
// Various vips  
pi_xtr         pi_inst;  
virtual pi_if.TB pi_port;  
host_xtr       host_inst;  
virtual host_if.TB host_port;  
ddr_xtr        ddr_inst;  
virtual ddr_if.TB ddr_port;
```

Start all agents

```
function build();  
    cfg.pi_cfg.pi_port = this.pi_port;  
    pi_inst = new( "sys_pi_xactor", "SYS_PI",  
                  0, cfg.pi_cfg);  
    cfg.host_cfg.host_port = this.host_port;  
    host_inst = new( "sys_host_xactor",  
                    "SYS_HOST", 0, cfg.host_cfg);
```

```
endfunction  
task start();  
    // Start up the rest of the transactors  
    `vmm_debug(this.log,"Started transactor  
    pi...");  
    pi_inst.start_xactor();  
    `vmm_debug(this.log,"Started transactor  
    host...");  
    host_inst.start_xactor();  
endtask  
start  
....
```



Customize Directory Structure

▶ `$SVF_HOME/templates/vmm_dir_struct/svf_dir_xml.tt`

XML tag (do not modify)

`<vmmdirs spec name="[% projName FILTER upper %]">`

`"MY_VMM_DIR"`

`<rootdir name="verif">`

`"my_verif"`

`<vipdir name="vips">`

`"my_vip"`

`<descr>` Directory containing all external and internal (VIPs)

description can be modified



Customized Directory Structure

SVF-TG Generated New Directory Structure

MY_VMM_DIR/
 my_verif/
 my_vip/

Project Directory
Verification database
Verification IPs

SVF-TG predefined XML tags for creating VMM compliant directory structure

- ▶ vmmdirspeg
- ▶ rootdir
- ▶ vipdir
- ▶ vipindir and more

[Reference: ***“SVF Template Generator Overview”*** from Paradigm-Works]



Customize File Templates

SVF-TG predefined **VMM-compliant** testbench templates

- Svf_proj_env_sv.tt
- Svf_proj_env_cfg_sv.tt
- Svf_vip_if_sv.tt
- Svf_vip_drv_sv.tt and more ...

[Reference: ***“SVF Template Generator Overview”***
from Paradigm-Works]



Template Syntax Examples

- ▶ There are two key components to a template file:
 - ▶ Text - This is the text that is reproduced as is in the generated output.
 - ▶ Directives- These are usually enclosed in [% %] and provide directions to the template processing code.

Examples of Directives :

```
[% INCLUDE header %]      # Include a file
[% IF numels <10 %]      # Conditional codes
    [% fileName %] has less than 10 entries
[% ELSE %]
    [% fileName %] has more than 10 entries
[% END %]
```



Example Template: *Svf_vip_sv.tt*

```
[%- vip = fileName | replace('.sv', '') -%]  
[%- Author = "Author:" -%]  
[%- File = "File:" -%]  
[%- Rev = "Revision:" -%]  
[%- Date = "Date:" -%]
```

**Assign values to variables.
User can change values**

```
/******  
*****
```

Use variables

```
* Author:    $[%- Author -%]$  
* File:      $[%- File -%]$  
* Revision:  $[%- Rev -%]$  
* Date:     $[%- Date -%]$
```

Text

```
*****  
*****
```

```
* Top level transactor wrapper file [% fileName %] for [% projName  
  %] verification environment
```

```
*****  
*****
```



Example Template: *Svf_vip_sv.tt*

```
`ifndef _[% vip FILTER upper %]_INC  
`define _[% vip FILTER upper %]_INC
```

User can change base class

**Make sure your template
is still VMM-compliant!!**

```
_xactor;
```

```
[% vip %]_cfg          cfg;  
[% vip %]_xmit        xmit;  
[% vip %]_rcv         rcv;
```

User can change component lists

```
virtual function void stop_xactor();  
    string msg;  
    super.stop_xactor();  
    $sformat(msg,"%s:stop_xactor()",get_instance());  
    `vmm_debug(log, msg);  
endfunction: stop_xactor  
endclass: [% vip %]  
`endif
```

User can change implementation



Conclusion

- ▶ 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

SystemVerilog FrameWorks™ Template Generator (Trial Version) is available freely for generating customized VMM environments.

Please contact

http: svf-tg.paradigm-works.com

email: 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™ 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 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:

