Verification Environment Automation from RTL

Zhidong Chen, Yunyang Song, Wenting Hou,
Junna Qiao, Junxia Wang, Ling Bai, Kei-Wang Yiu
MediaTek, Inc.

Email: zhidong.chen@mediatek.com

Abstract

SoC Scale Increases

→ Complex Verification Environment

- E.g., hundreds of interfaces in DUT

→ **Verification Environment Automation Is A Must**

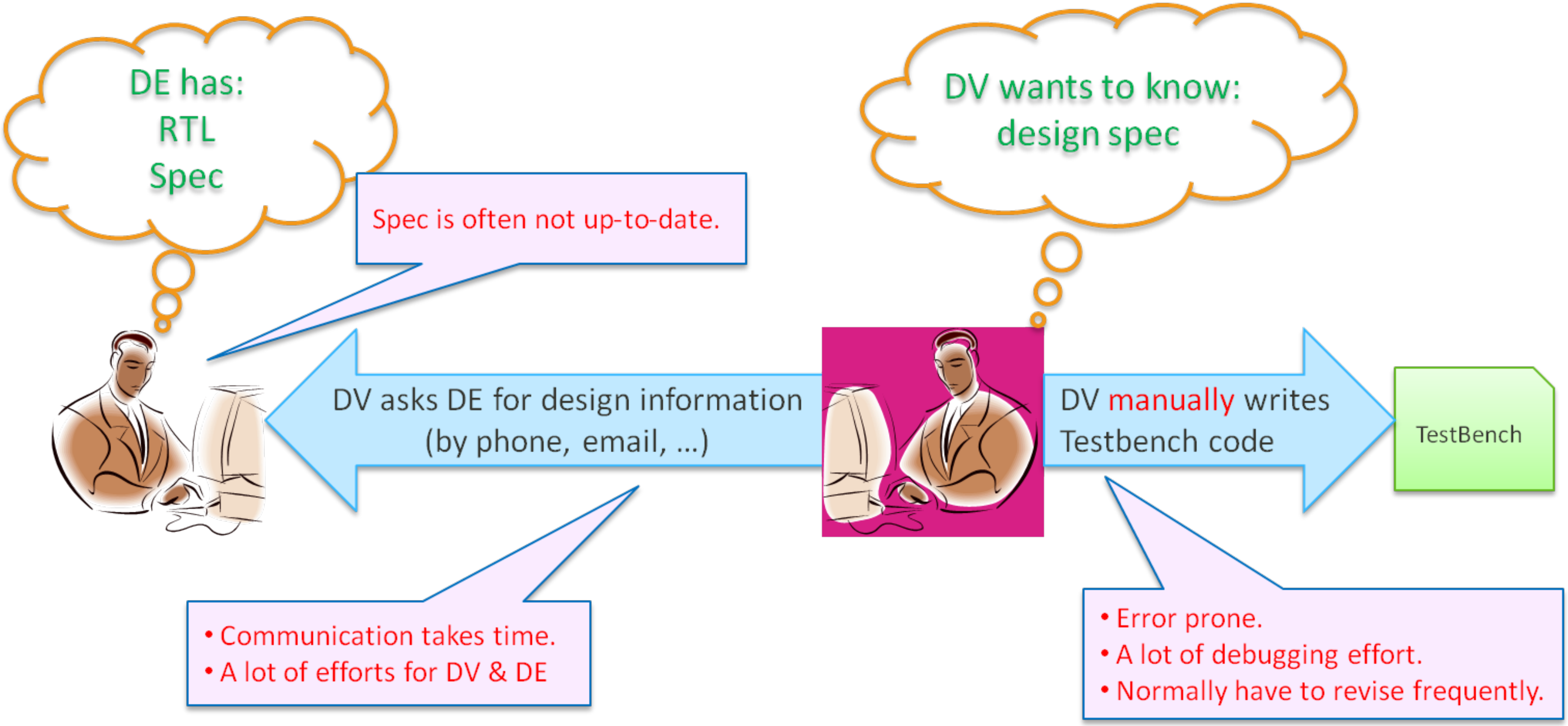
- We propose **a solution that can automatically build a verification environment from RTL**.
  - Deployed in real projects
  - **Manual effort reduced by ~70%**

Motivation

SoC Scale Increases

→Coordination (DV vs. designer) is **painful** & **time-consuming**

→**Too much manual effort** in building a verification environment



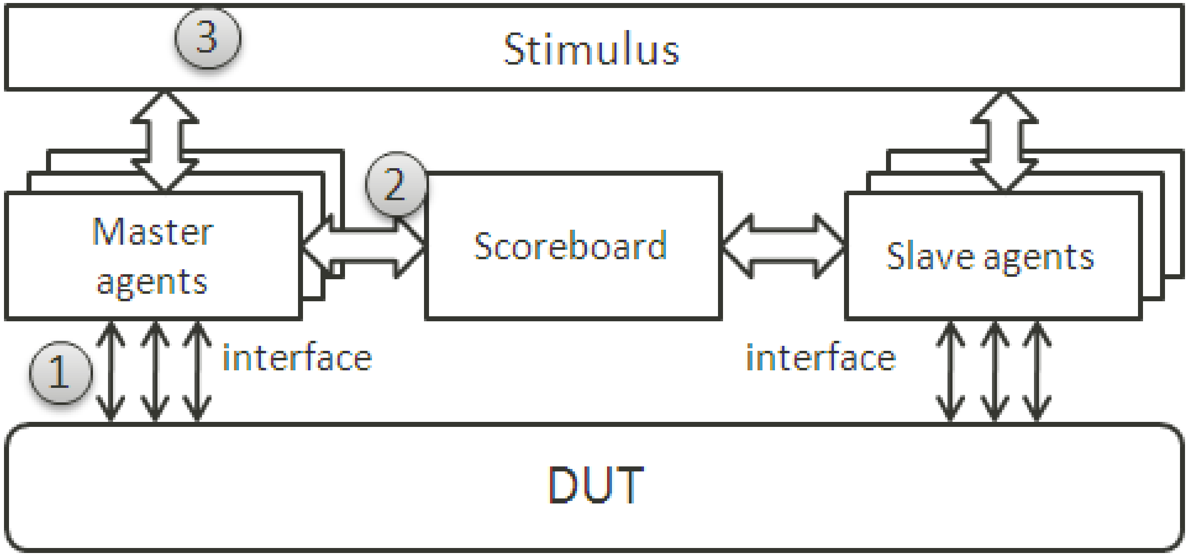
DE has:
RTL
Spec
DV wants to know:
design spec
DV asks DE for design information (by phone, email, …)
DV manually writes Testbench code
TestBench

Spec is often not up-to-date.

- Communication takes time.
- A lot of efforts for DV & DE

- Error prone.
- A lot of debugging effort.
- Normally have to revise frequently.

Bus Fabric Verification Challenge

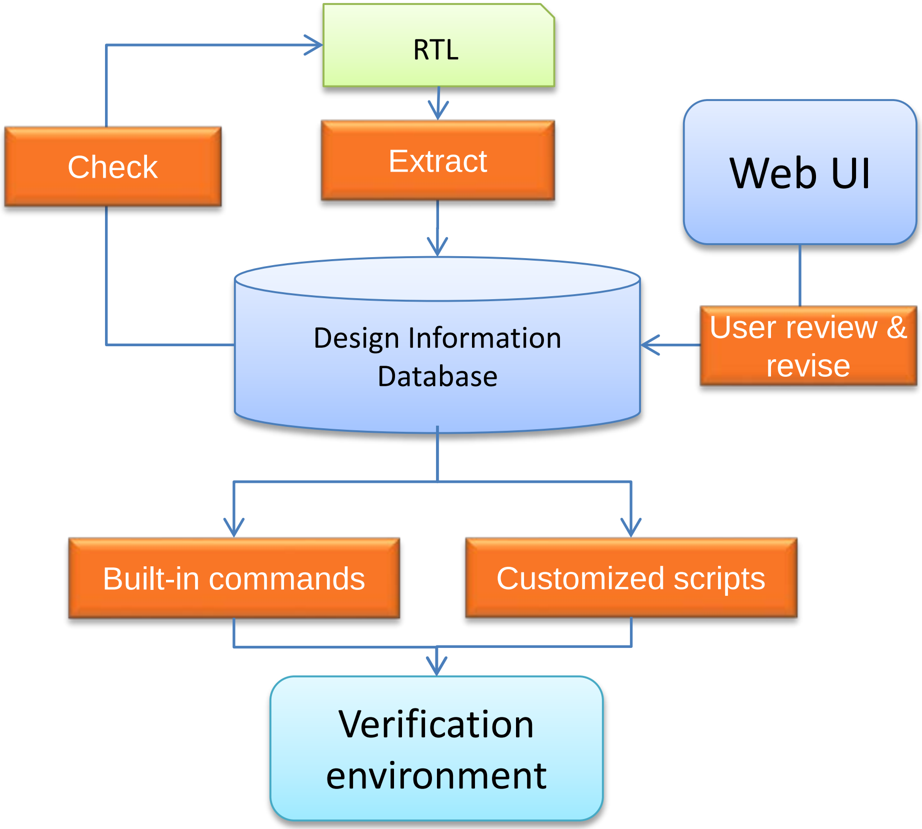


Category	Challenges
VIP Interface Connection	• ~6 protocol types • > 180 interfaces • > 4200 signal connections
VIP Configuration	• AXI master VIP has >20 configurations • > 1800 configurable variables for all VIPs
VIP Stimulus Constraint	• Need to customize transaction constraints.

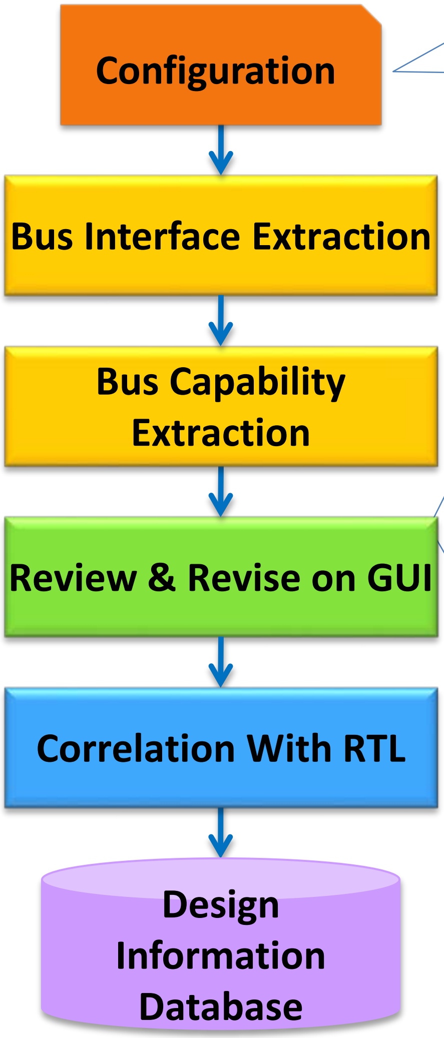
Solution: Verification Environment Automation from RTL

We provide a solid solution to

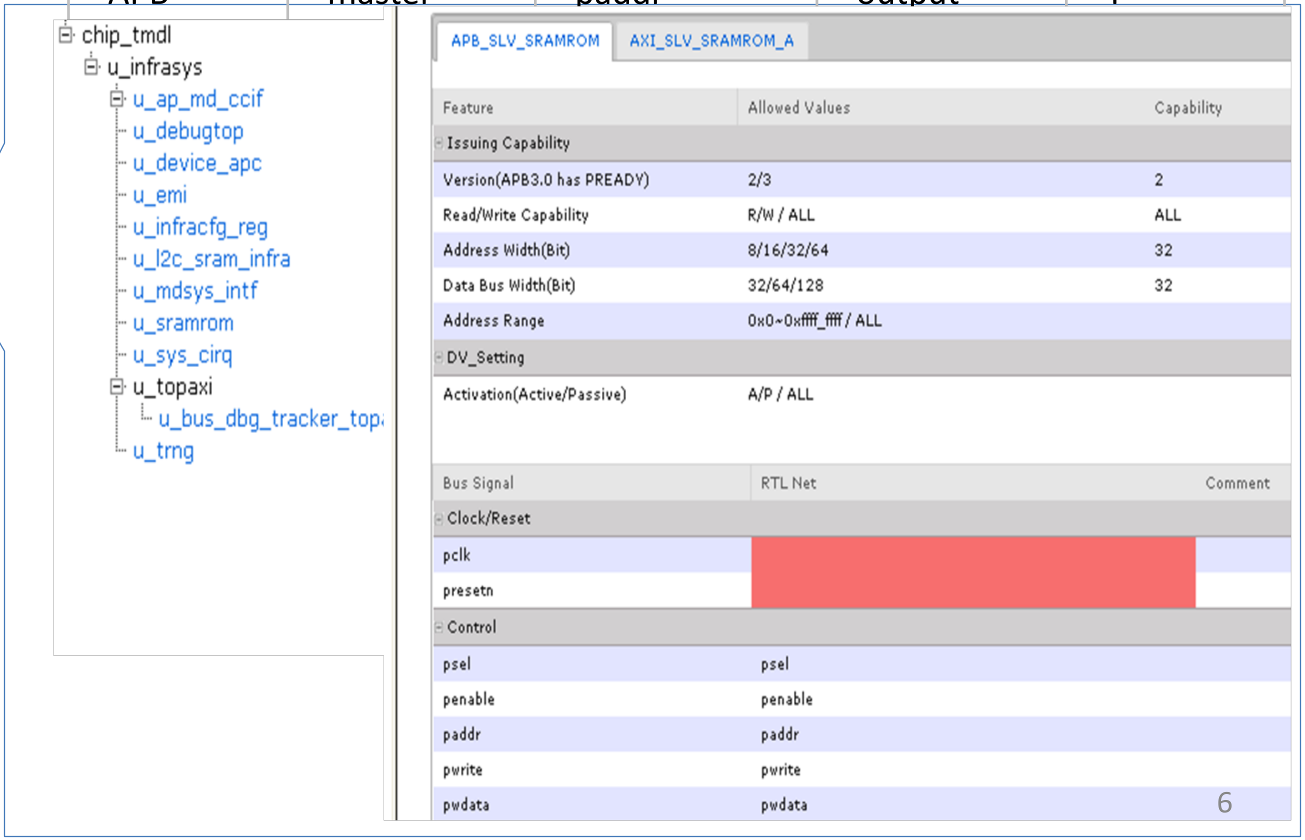
- Extract design information from RTL
- Provide an easy-to-use GUI for the users to review and input
- Check the acquired design information against RTL
- Automatically build the verification environment based on the design information



Extract Design Information from RTL



#Protocol	Type	Signal	Direction	Seed
APB	master	pclk	input	N
APB	master	presetn	input	N
APB	master	penable	output	N
APB	master	paddr	output	Y



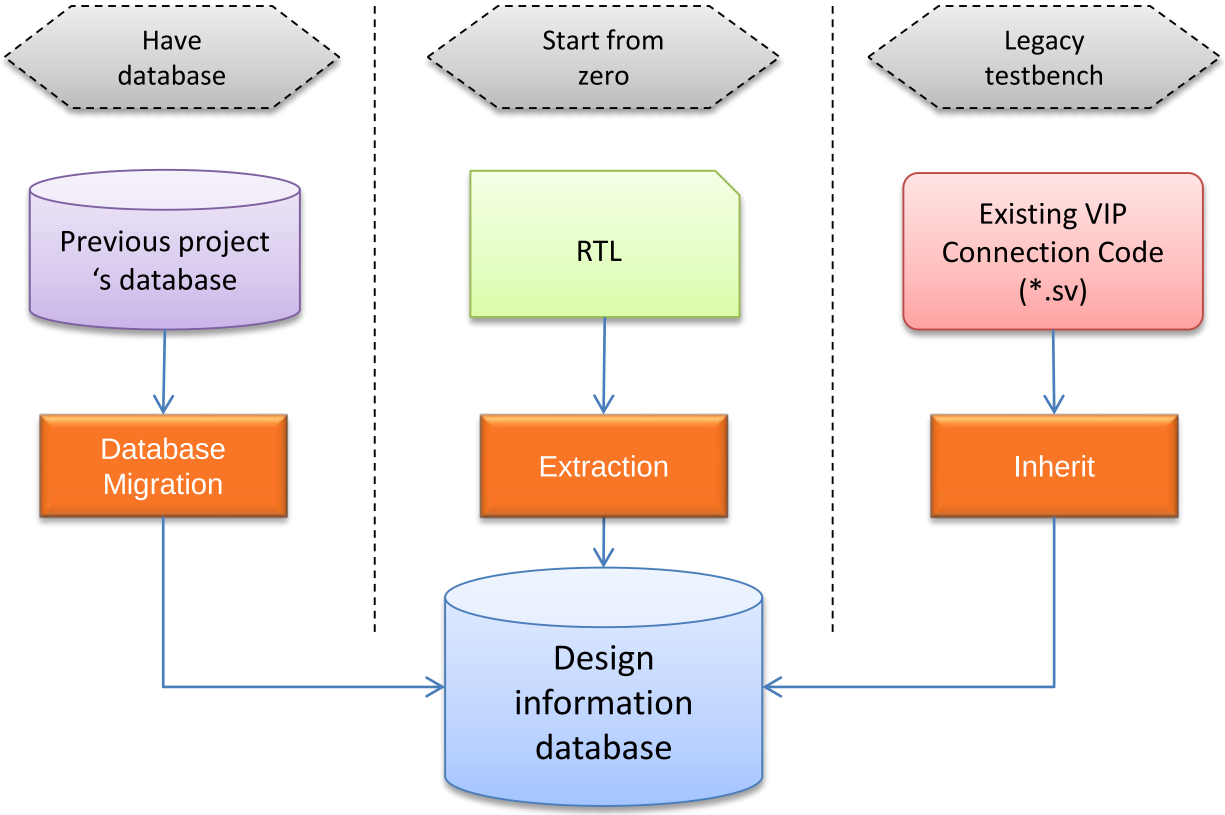
- chip_tmdl
  - u_infrasys
    - u_ap_md_ccif
    - u_debugtop
    - u_device_apc
    - u_emi
    - u_infracfg_reg
    - u_l2c_sram_infra
    - u_mdsys_intf
    - u_sramrom
    - u_sys_cirq
    - u_topaxi
      - u_bus_dbg_tracker_top
    - u_trng

Feature	Allowed Values	Capability
Issuing Capability		
Version(APB3.0 has PREADY)	2/3	2
Read/Write Capability	R/W / ALL	ALL
Address Width(Bit)	8/16/32/64	32
Data Bus Width(Bit)	32/64/128	32
Address Range	0x0~0xffff_ffff / ALL	
DV_Setting		
Activation(Active/Passive)	A/P / ALL	

Bus Signal	RTL Net	Comment
Clock/Reset		
pclk		
presetn		
Control		
psel	psel	
penable	penable	
paddr	paddr	
pwrite	pwrite	
pwdata	pwdata	

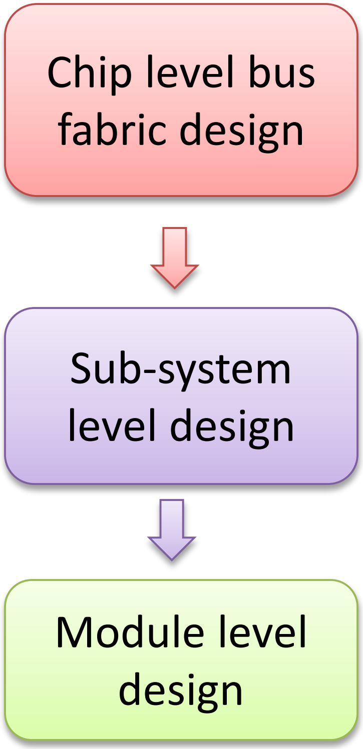
We can also get information by …

- **Database migration**
  - Copy and revise the database from previous designs.
- **Inherit from legacy testbench**
  - Extract VIP connections from existing testbench codes


Have database
Start from zero
Legacy testbench
Previous project 's database
RTL
Existing VIP Connection Code (*.sv)
Database Migration
Extraction
Inherit
Design information database

Verification Environment Build-up

Testbench automation can be applied to designs of various scales.



Chip level bus fabric design

- >180 bus interfaces
- Bus information extraction + manually review
- VIP Interface connection, configuration, transaction constraint

Sub-system level design

- Sub-system bus TBA (~30 bus)
- User needs to write simple scripts (<50 lines)
- > 60% of the testbench can be automatically generated.

Module level design

- Module level testbench
- Push button solution
- > 80% of the testbench can be automatically generated.

Verification Environment Build-up (Module level design)

- **Push-button solution**

  As the design is relatively simple, we have simplified the above flow into a push-button solution for module-level verification.

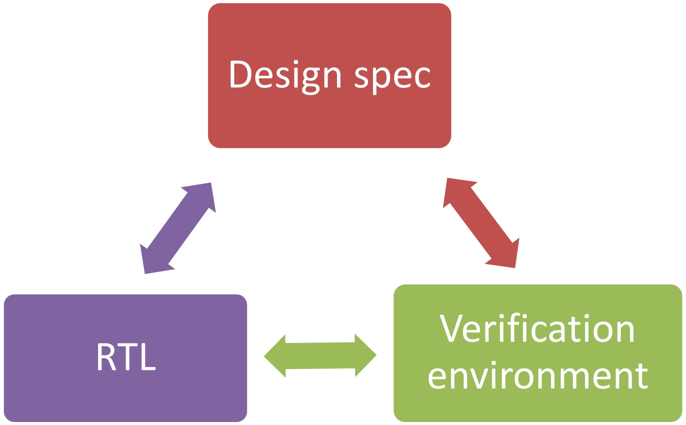
- **Generated templates (based on UVM)**

Generated templates	Refinement needed?	Comment
Testbench top file	NO	Including DUT instantiation, VIP agent instantiation
VIP interface connection	YES(if necessary)	Some signals do not follows bus naming rule, e.g., clock and reset
VIP Configuration	YES	Some configuration can't be extracted from RTL, e.g., AXI max outstanding capabilities
Scoreboard template	NO	TLM connections from VIP agents' subscribers to scoreboard components
Customized transaction class	YES(if necessary)	Only if the DUT is not full functioning
Function coverage	NO	Auto-generated base on bus capabilities
Script for running simulations	NO	-

Verification Environment Build-up (Sub-system/Whole chip)

Cross-checking with design specifications

- **Memory map**
  - Interface locations
  - Address decoding information



Design spec

RTL

Verification environment

Region Name as a unique ID

Protocol and bus location are used to identify bus interfaces in RTL.

Region Name	Start Address	Size	Protocol	Bus Location
SRAM	0x0000_0000	0x1000_0000	AXI	`TOP.u_sram0
USB0	0x1000_0000	0x1001_0000	AHB	`TOP.u_usb0
USB1	0x1001_0000	0x0001_0000	AHB	`TOP.u_usb0
SPI	0x1002_0000	0x0001_0000	APB	`TOP.u_spi0
DRAM_BANK1	0x8000_0000	0x1000_0000	AXI	`TOP.u_dram0
DRAM_BANK2	0x9000_0000	0x1000_0000	AXI	`TOP.u_dram1

Experimental Results

- **Speed up testbench stabilization**
  - First pattern regression passing reduced from 9 weeks to 3 weeks (reduced ~65%)
  - Average testbench file revision is reduced from 11.1 to 3.4~3.6 (reduced ~70%)

	Previous	Now
Information Collection	3 weeks	< 1 week
Testbench Generation	2 weeks	1 day
Environment Stabilization	4 weeks	2 weeks
Total	9 weeks	3 weeks

- **[Faster building-up + Faster iteration] = Faster regression and coverage closure**

	Bus interface	Bus interface type	Regression pass (day)	Coverage closure (day)
Single module	10	2	0.5	2
Sub-system	>30	5	4	8
Whole chip	>180	6	20	-

Summary

- Design information can be **auto-extracted** from RTL with **limited configurations**.
- **Configurable** bus protocol, including both standard bus (AMBA, etc.) and user-defined bus.
- **Easy-to-use GUI** for design information **review and revision**.
- Design information can be **inherited from existing testbench and database migration**.
- Full spectrum support for **different design complexity**, from module level to SoC level design.
- **Correlation with RTL and design specification**, relieving DV from tedious debugging work.
- Verification environment building-up time for SoC designs are **reduced from months to weeks.**

Contact Information

MediaTek (Beijing) Inc.

Building 1-B, No. 6 Park,
Jiuxianqiao Road,
Chaoyang District, Beijing,
China 100015

Telephone:
+86-10-5690-0888

Email:
zhidong.chen@mediatek.com