



# A Tool to Aid to Generate Adaptive MBIST Test Vectors End to End

Weili Wang([weilwang@cisco.com](mailto:weilwang@cisco.com))

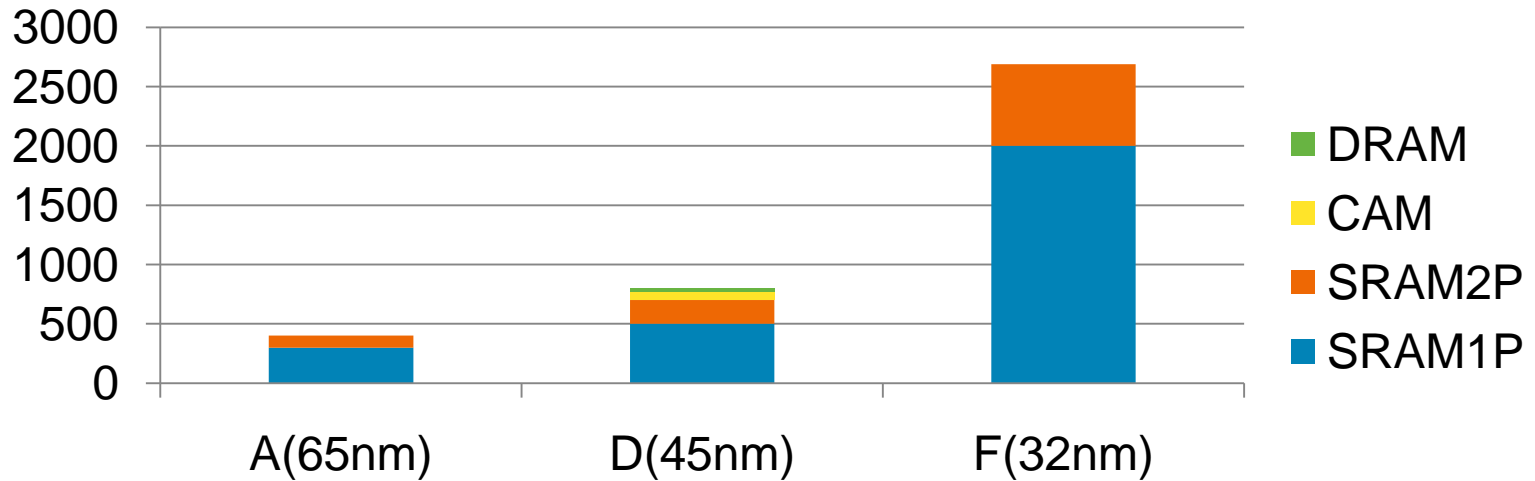
# Background

- ❑ Heat dissipation issue when run at-speed MBIST all at once with all clocks on, which is different from mission mode low power design.
- ❑ Memory count increasing dramatically.
- ❑ Need to configure many control/observe registers to run MBIST.
- ❑ Serialized hardware access interface to resolve routing congestion.
- ❑ Need programmable mechanism to optimize Time/Power/Granule.
- ❑ In [4] and [5], programmable MBIST proposed, but not accessed by enumeration, neither details about end-to-end applications.

Tests	ATE	JTAG	CPU	Time	Power	Granule
Wafer	√			Critical		
Package	√	√		Critical		
Board		√			Critical	Maybe
Pre-2-conner		√	√		Critical	Maybe
2-conner		√	√		Critical	Maybe
In-field			√		Critical	Maybe
diagnostic	√	√				Critical

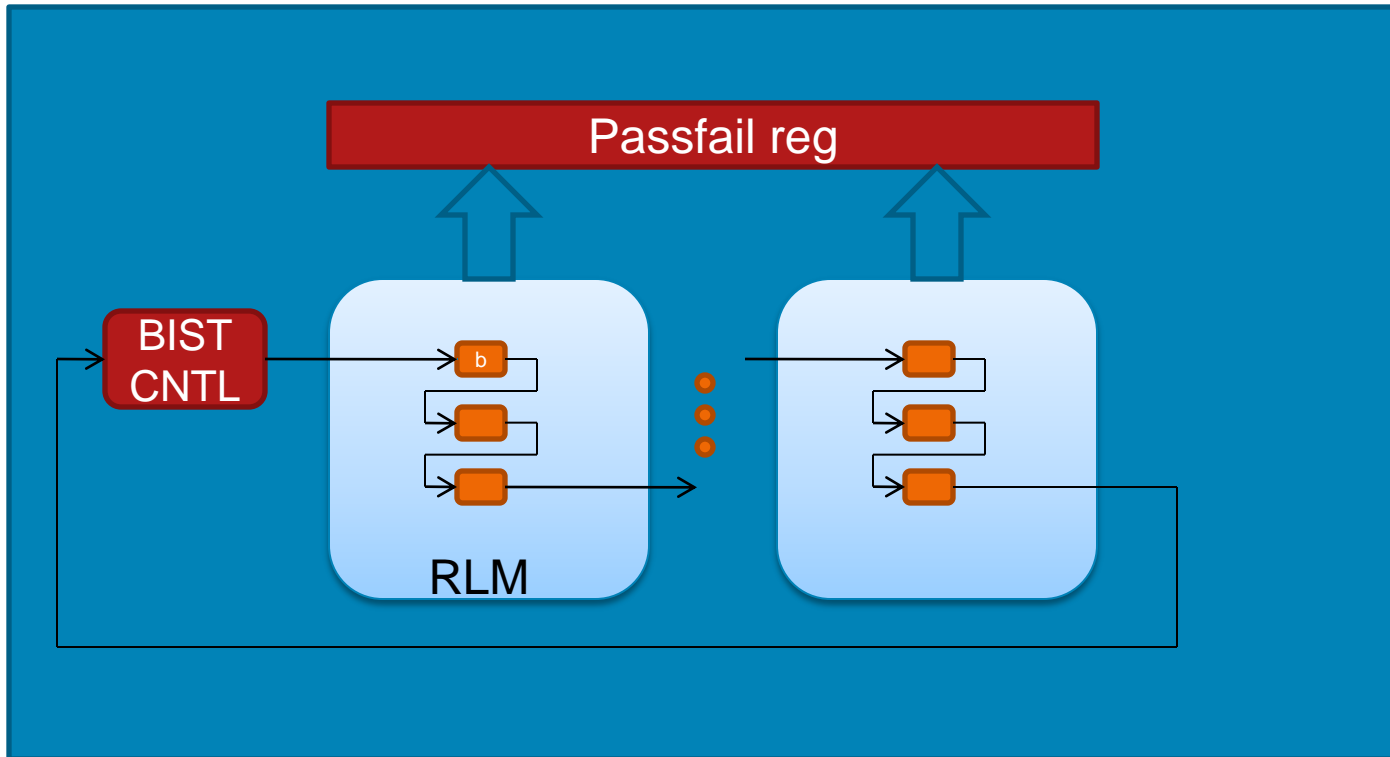
## End-to-end tests and their specific requirements

# Memories count / control regs



Regs to control	Memory	Address	Pattern	Margin
SRAM1P	8'b	15'b	15'b	4'b
SRAM2P	8'b	14'b	20'b	4'b
CAM	8'b	11'b	30'b	4'b
DRAM	4'b	20'b	20'b	4'b

# Hardware Diagram

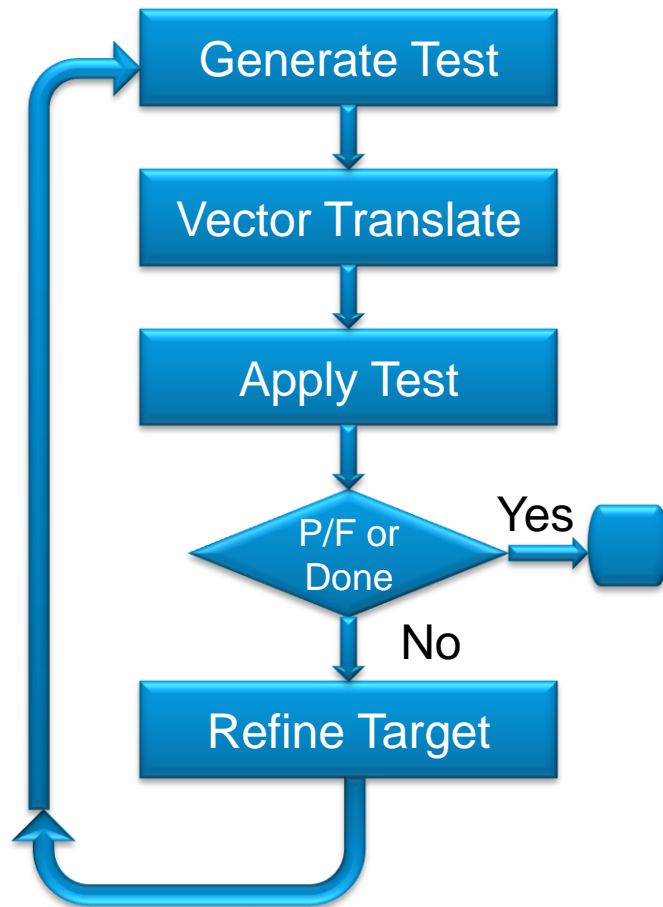


All BISTs are chained and assigned a software ID

All PASSFAILs are monitored by registers

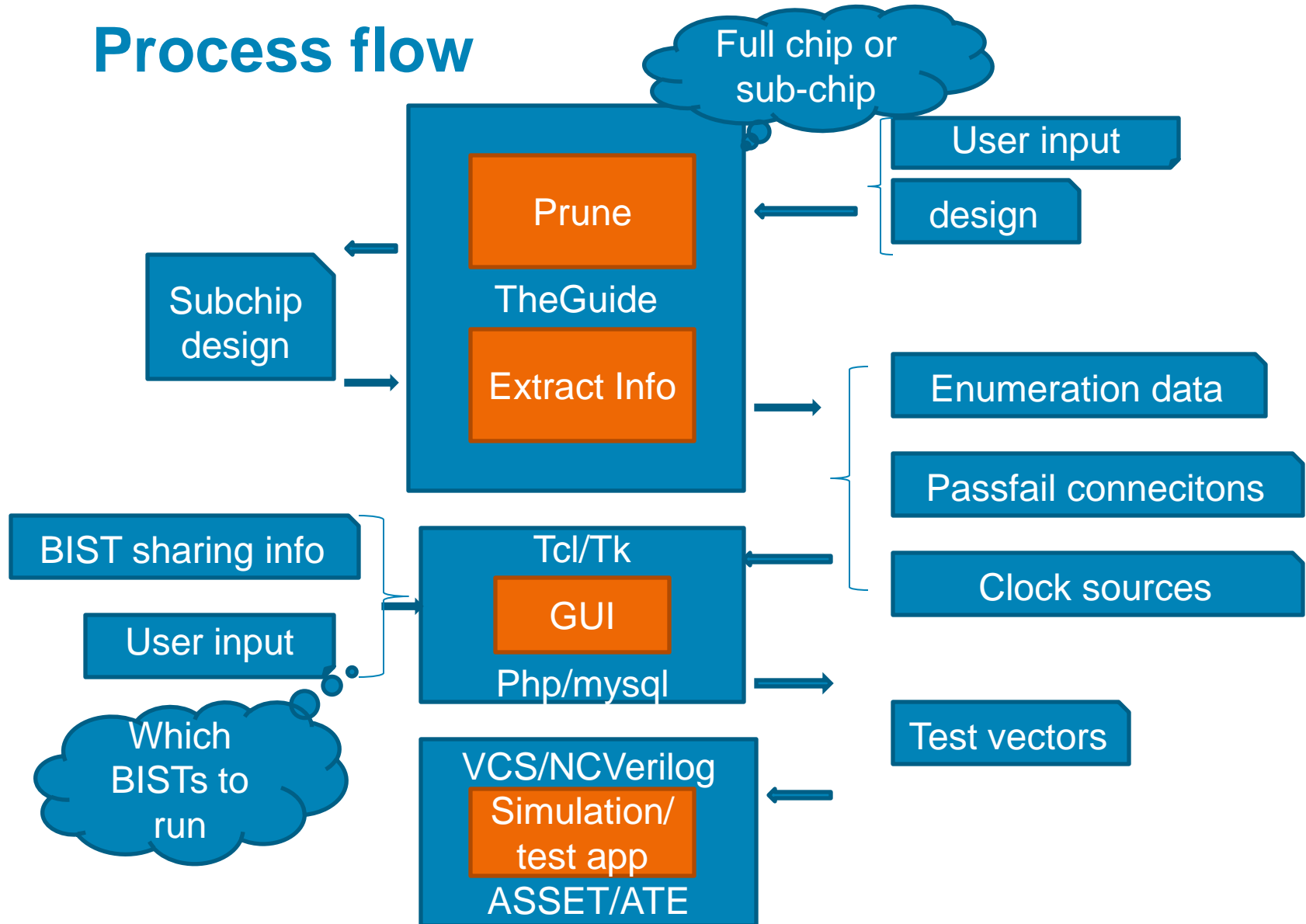
Refer to [3], about BISTCNTL VMAC

# Test access / control flow chart



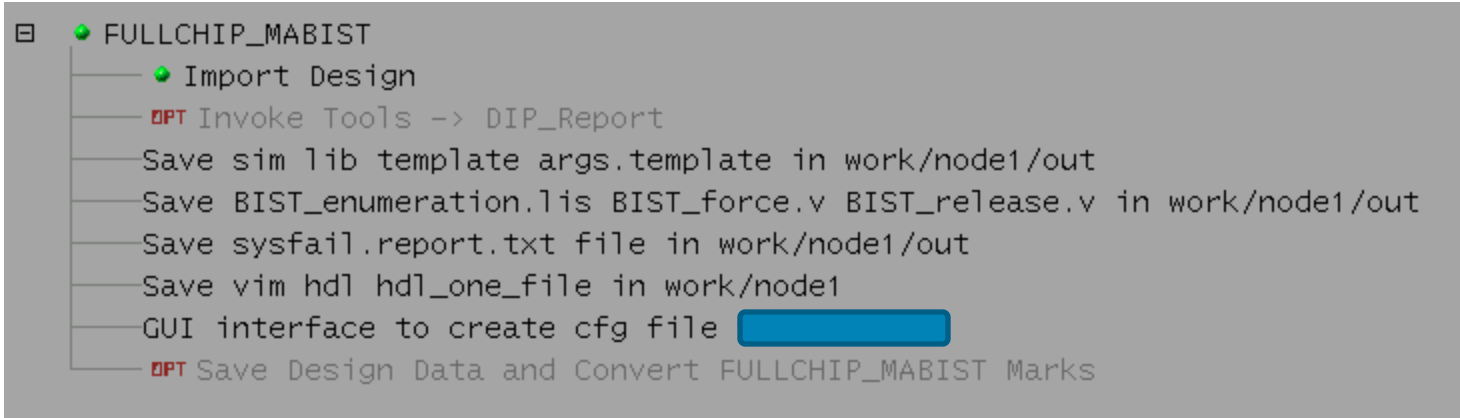
- Pre-amble (Power on reset, PLL lock, TDR control)
- Enumeration (Assign an ID for each BIST)
- Decide which BISTs to run
- Decide which Memories to run
- Decide Address range
- Decide which Patterns
- Launch BIST run
- Monitor BIST Done
- Monitor BIST PassFail
- Monitor Memory PassFail

# Process flow



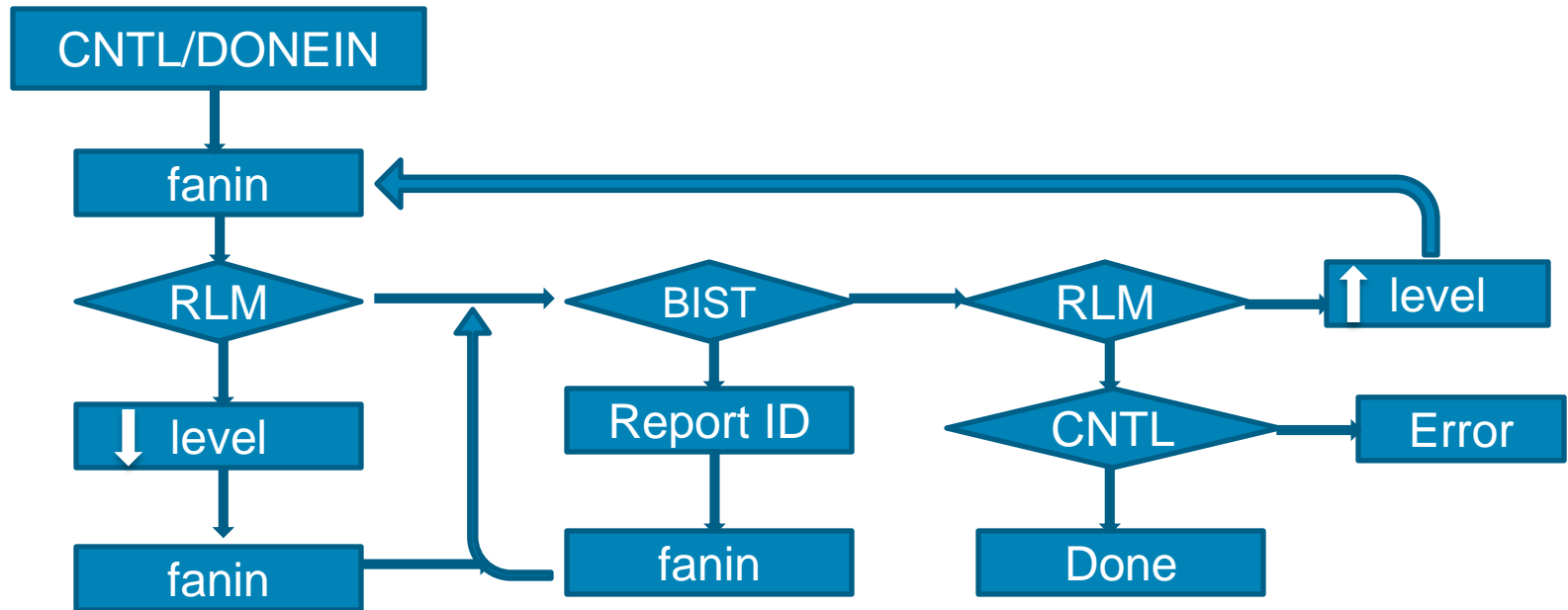
# GUI for design structure information

Built inside IBM TheGuide Platform for process control



```
<methodology>
<process name="FULLCHIP_MABIST">
  <task text="FULLCHIP_MABIST"
    doc="FULLCHIP_MABIST.html">
    <tasklist type="sequential">
      <task text="Import Design"
        label="ImportDesign"
        doc="fepmeth/FEP_DFT_import_design.html"
        command="FEP_DFT_design_import -overrides {KEY FEP_Part1_[gna::get_current_proto]}" >
      </task>
      <task text="Save BIST_enumeration.lis BIST_force.v BIST_release.v in work/node1/out"
        label="enumerate_bist"
        command="BooleDozer_Command_Via_Socket -key {FEP_Part1_[gna::get_current_proto]} -overrides {save_command
at::tg_fep_checkpoint command {source $gna::design_setup(working_dir)/../tcl/FEP_ST_prn_enumerate_bist}}" >
      </task>
    </tasklist>
  </task>
</process>
</methodology>
```

# Extract enumeration ID process/code



```
set fanin [led::all_fanin -no_only_cell -to $net];  
set inst [file dirname $fanin];  
set rlm [led::get_parent -port $fanin];  
idm::set_active_proto -def $rlm -view TECH set rlm_fanin  
[led::all_fanin -no_only_cell -to [file tail $fanin]];  
set bist [file dirname $rlm_fanin];
```



# Extracted design data

ID	Instance name	Type
1	Sc12.Sec/SecCore/SecPartD/SecReplayWindow/GenericMem/gen_U1/mem_0_0/mem/BIST	BIST1SPBHCH
2	Sc12.Sec/SecCore/SecPartE/SecEgrMem/GenericMem/gen_U1/mem_0_0/mem/BIST	BIST2SPBHCH
3	Sc12.Sec/SecCore/SecPartD/SecStatistics/SecStatistics/GenericMem/gen_U1/mem/BIST	BIST1SPBHCH
4	Sc12.Sec/SecCore/SecSaTable/SecSAEntryTable/GenericMem/gen_U1/mem/BIST	BISTDRAMA

## List enumeration ID

```

SYSFAIL[0] pin = Sc0/Aqm/AqmCpp/AqmCppVirtualBinTable/GenericMem/gen_U1/mem/BIST/SYSFAIL0
SYSFAIL[1] pin = Sc0/Aqm/AqmCpp/AqmCppVirtualBinTable/GenericMem/gen_U1/mem/BIST/SYSFAIL1
SYSFAIL[2] pin = Sc0/Aqm/AqmCpp/AqmCppVirtualBinTable/GenericMem/gen_U1/mem/BIST/SYSFAIL2
SYSFAIL[3] pin = Sc0/Aqm/AqmCpp/AqmCppVirtualBinTable/GenericMem/gen_U1/mem/BIST/SYSFAIL3

```

## List fail net connections

```

{
  Sc11
  Hsh0/HshTableAd/GenericMem/gen_U1/mem/WRAP3_DRAM #DRAMC08X0256X8X1X146V10M2C1 MEMCLK=sysClk
  Hsh1/HshTableAd/GenericMem/gen_U1/mem/WRAP3_DRAM #DRAMC08X0256X8X1X146V10M2C1 MEMCLK=sysClk
  Hsh4/HshTableAd/GenericMem/gen_U1/mem/WRAP3_DRAM #DRAMC08X0256X8X1X146V10M2C1 MEMCLK=sysClk
}

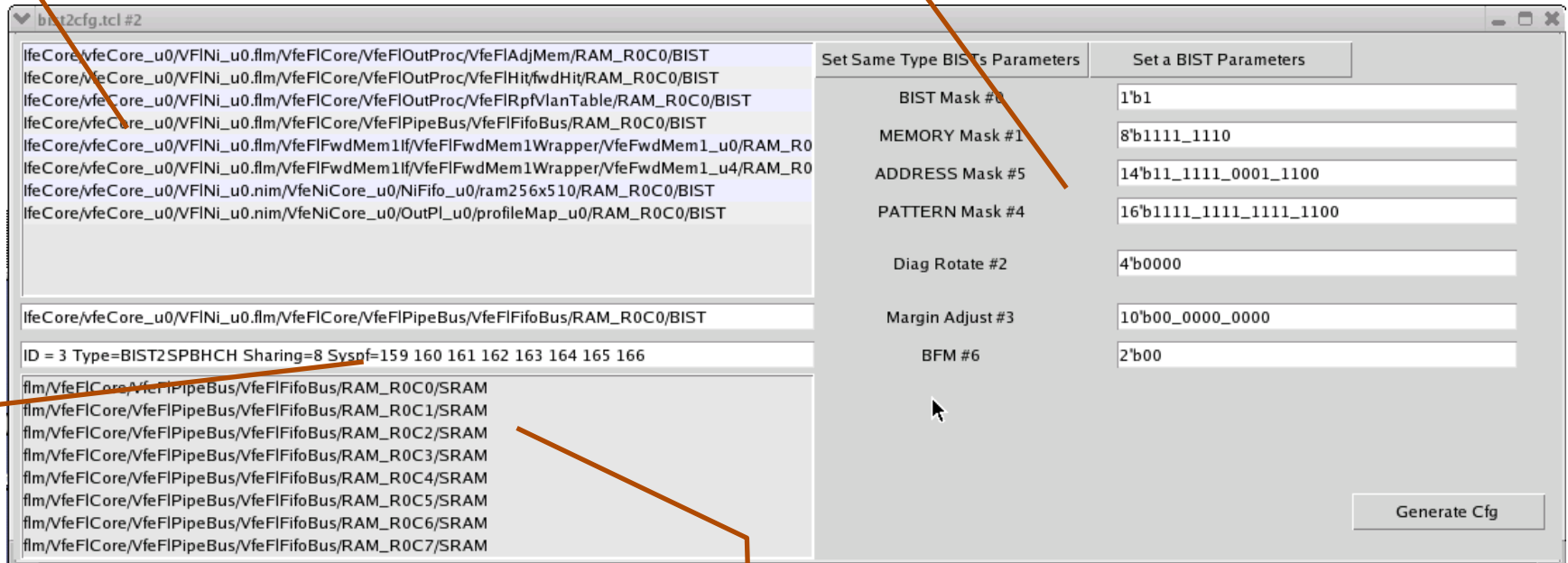
```

## List clock source

# GUI: Test Configuration Generation

# Enumeration data

# Control registers

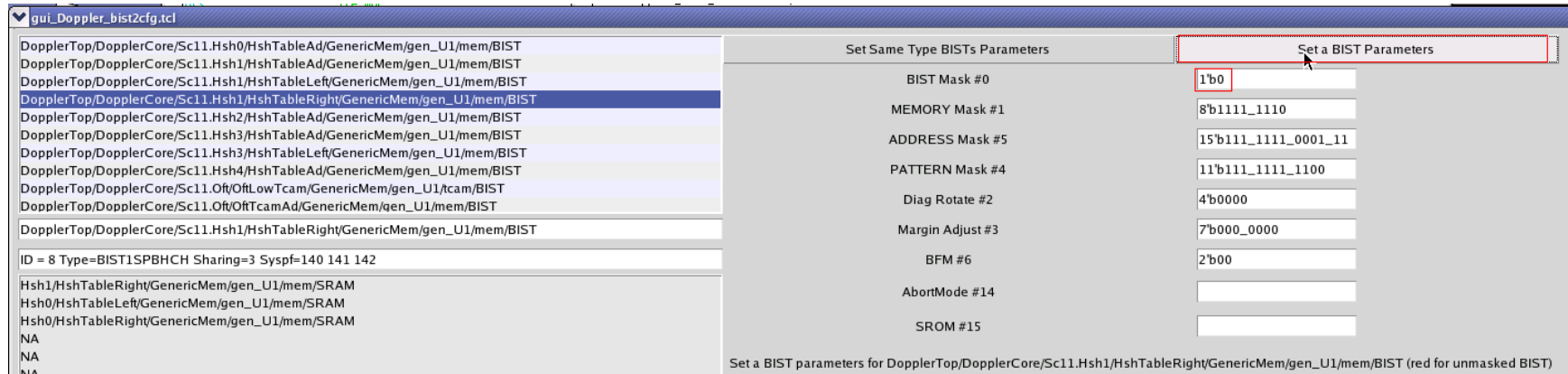


# Passfail connctions

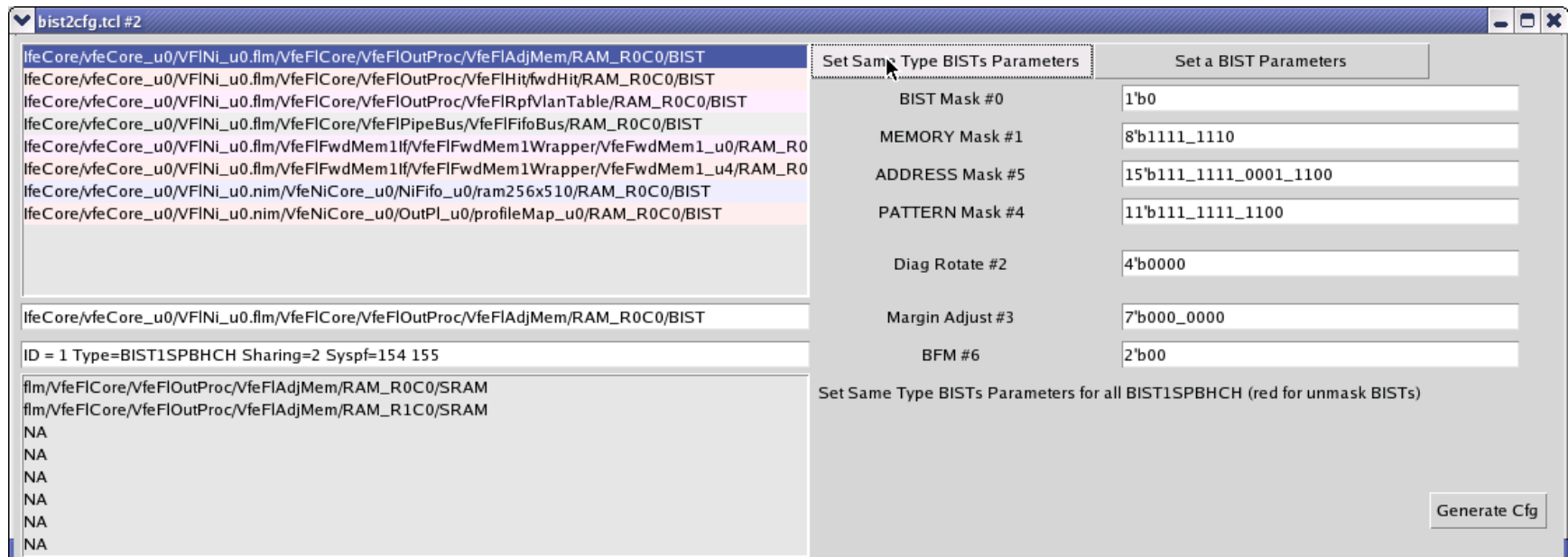
## BIST sharing data

# Choose which BISTs to run

Click on a BIST to set its parameter; It will turn **red** when Mask=0



Set parameters for All BISTs with the same type.



# Example test configuration file (control)

```
#BSHCNTL WRITE BIST 1 for BISTMASK
# =====
# Write BIST 1 CONTROLREG0 BISTMASK
# =====
INSTR BSHCNTL 111111111111101111111110
SHIFTDR 20 {
  OPS_WRITE      3 3'b110 # IREG[17-19]
  TYPE           1 1'b0 # IREG[16]
  ADDRESS        16 16'h0800 # IREG[0-15] reverse10
}
RUNTEST 16
# BSHDATA write BISTMASK 1
INSTR BSHDATA 111111111111101111111101
SHIFTDR 32 {
  BISTMASK 32 32'h80000000 # 1-bit BISTMASK
}
RUNTEST 16
```

Instruction

Data

bistmask

```
# BSHCNTL WRITE BIST 2 for MEMORYMASK
# =====
# Write BIST 2 CONTROLREG0 MEMORYMASK
# =====
INSTR BSHCNTL 111111111111101111111110
SHIFTDR 20 {
  OPS_WRITE      3 3'b110 # IREG[17-19]
  TYPE           1 1'b0 # IREG[16]
  ADDRESS        16 16'h8400 # IREG[0-15] reverse21
}
RUNTEST 16
# BSHDATA write MEMORYMASK 1
INSTR BSHDATA 111111111111101111111101
SHIFTDR 32 {
  MEMORYMASK 32
  01111111100000000000000000000000
  #for SRAM0 SRAM1 to SRAM7
}
RUNTEST 16
```

memorymask

# Example test configuration file (status)

```
# BSHCNTL READ
INSTR BSHCNTL 11111111111110111111110
SHIFTD 20 {
OPS_READ      3 3'b010 # IREG[17-19]
TYPE          1 1'b0 # IREG[16]
ADDRESS       16 16'h2800 # IREG[0-15] reverse
              14
}
RUNTEST 16
# BSHDATA read done result
INSTR BSHDATA 111111111111101111111101
SHIFTD 160 {
# Total 4 BISTs => reverse of 12'h200
BISTFAIL_BISTCNTL 32 32'h0 (32'h200000003)
BISTFAIL4 32 32'h0 (32'h000000000)
BISTFAIL3 32 32'h0 (32'h000000000)
BISTFAIL2 32 32'h0 (32'h000000000)
BISTFAIL1 32 32'h0 (32'h000000000)
}
RUNTEST 16
```

```
INSTR MABIST 111111111111110111111111
RUNTEST 16
```

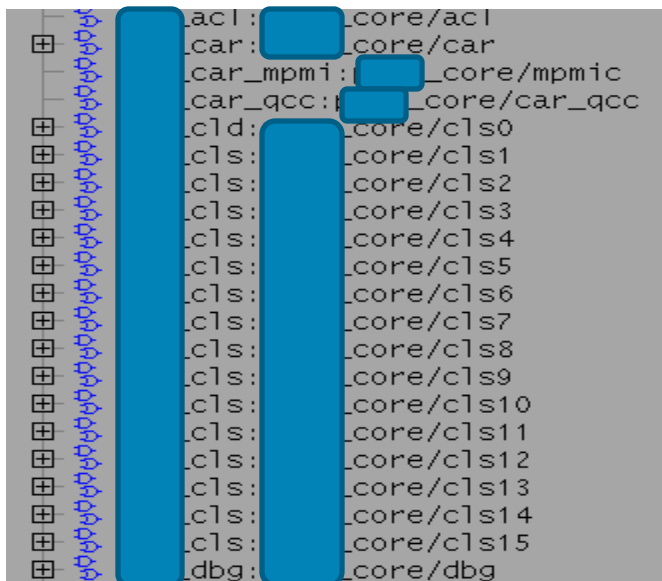
```
SHIFTD 842 {
sys_mabist_observe_tdr.SYSPF0 1 1 (1)
sys_mabist_observe_tdr.SYSPF1 1 1 (1)
sys_mabist_observe_tdr.SYSPF2 1 1 (1)
sys_mabist_observe_tdr.SYSPF3 1 1 (1)
sys_mabist_observe_tdr.SYSPF4 1 1 (1)
sys_mabist_observe_tdr.SYSPF5 1 1 (1)
sys_mabist_observe_tdr.SYSPF6 1 1 (1)
sys_mabist_observe_tdr.SYSPF7 1 1 (1)
sys_mabist_observe_tdr.SYSPF8 1 1 (1)
sys_mabist_observe_tdr.SYSPF9 1 1 (1)
```

Embedded ID to help to identify defected RAMs

# Sub-chip: manageable simulation

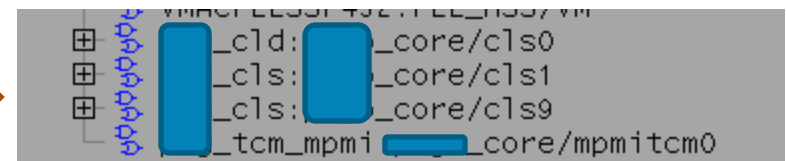
```
PRUNE_MABIST
  Import Design
  Create select_rlm_to_delete.tcl
  Edit supp_select_rlm_to_delete.tcl for keeping instances before running
  Run PRUNE script
  DPT removerEGs for MABIST sim
  DPT Invoke Tools -> DIP_Report
  Save sim lib template args.template in work/node1/out
  Save BIST_enumeration.lis BIST_force.v BIST_release.v
  Save vim_prune hdl_prune hdl_prune_one_file in work/node1
  GUI interface to create cfg file
  DPT Save Design Data and Convert PRUNE_MABIST Marks
```

List all RLM  
instances for  
the user to  
choose



```
act: core/act
car: core/car
car_mpmi: core/mpmic
car_qcc: core/car_qcc
cls: core/cls0
cls: core/cls1
cls: core/cls2
cls: core/cls3
cls: core/cls4
cls: core/cls5
cls: core/cls6
cls: core/cls7
cls: core/cls8
cls: core/cls9
cls: core/cls10
cls: core/cls11
cls: core/cls12
cls: core/cls13
cls: core/cls14
cls: core/cls15
dbg: core/dbg
```

A much smaller net-list for simulation



```
vim: core/vim
cls: core/cls0
cls: core/cls1
cls: core/cls9
tcm_mpmi: core/mpmitcm0
```

# Simulation run time / log

	fsdb size	Run time		
		compile	enum	Run bist
fullchip	Too big	4 hrs	14 days	Too long
sc1	15MB	20 mins	4 hrs	10 hrs
sc2	20MB	20 mins	4 hrs	10 hrs

Time 924840.00 NS : Simulating (current) JTAG Vector = 9250 ....  
 Time 926840.00 NS : Simulating (current) JTAG Vector = 9300 ....  
 Time 927880.00 NS : Failed vector# **9325** (@CYL= 1): Tdo= 'x' -- Expected ('0')  
 Time 929440.00 NS : Simulating (current) JTAG Vector = 9350 ....

log

9323 11000 // [156] BISTFAIL1 [28].TDI => 0 BISTFAIL1 [29].TDO => 0  
 9324 11000 // [157] BISTFAIL1 [29].TDI => 0 BISTFAIL1 [30].TDO => 0  
**9325** 11000 // [158] BISTFAIL1 [30].TDI => 0 BISTFAIL1 [31].TDO => 0  
 9326 1110X // [159] BISTFAIL1 [31].TDI => 0 ShiftDR => Exit1DR  
 9327 1111X // Exit1DR => UpdateDR

Test vector

BIST ID

# Conclusion

- TheGuide platform provides XML/TCL/TK for easy script integration and methodology management.
- Automated scripts extract design structure information to help to generate test configuration.
- GUI help to target specific memories to run for power, run time, or diagnostic need.
- Embed the memory ID in the test configuration file and test vectors helps diagnostics.
- Much smaller sub-chip net-lists enable quick simulation verification closure.



# Acknowledgement

- Thank Douglas Pricer and Valerie H Chickanosky at IBM for technical support.

## Reference

- 1) TheGuide User's Guide , <https://www-309.ibm.com/technologyconnect>
- 2) Tcl Developer Xchange, <http://www.tcl.tk>
- 3) Cu-65HP BISTCNTL Virtual Macro (VMAC) Databook, <https://www-309.ibm.com/technologyconnect>
- 4) Ismet Bayraktaroglu, Olivier Caty, Yickkei Wong, "Highly Configurable Programmable Built-In Self Test Architecture for High-Speed Memories," vts, pp.21-26, 23rd IEEE VLSI Test Symposium (VTS'05), 2005
- 5) Chung-Fu Lin ; Chia-Fu Huang ; De-Chung Lu ; Chih-Chiang Hsu ; Wen-Tsung Chiu ; Yu-Wei Chen ; Yeong-Jar Chang ; "A Low-Cost Programmable Memory BIST Design for Multiple Memory Instances," Test Conference, 2008. ITC 2008. IEEE International



**CISCO**