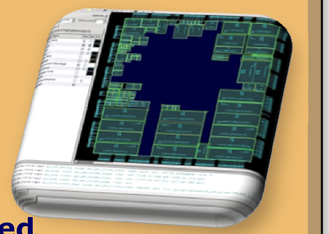
An RTL Approach to Memory-BIST Insertion with Proprietary Architectures

Purpose and Motivation

- Replace the memories in a design at RTL or gates with BIST components
- Automate the BIST insertion process while ensuring that the design functionality is not compromised

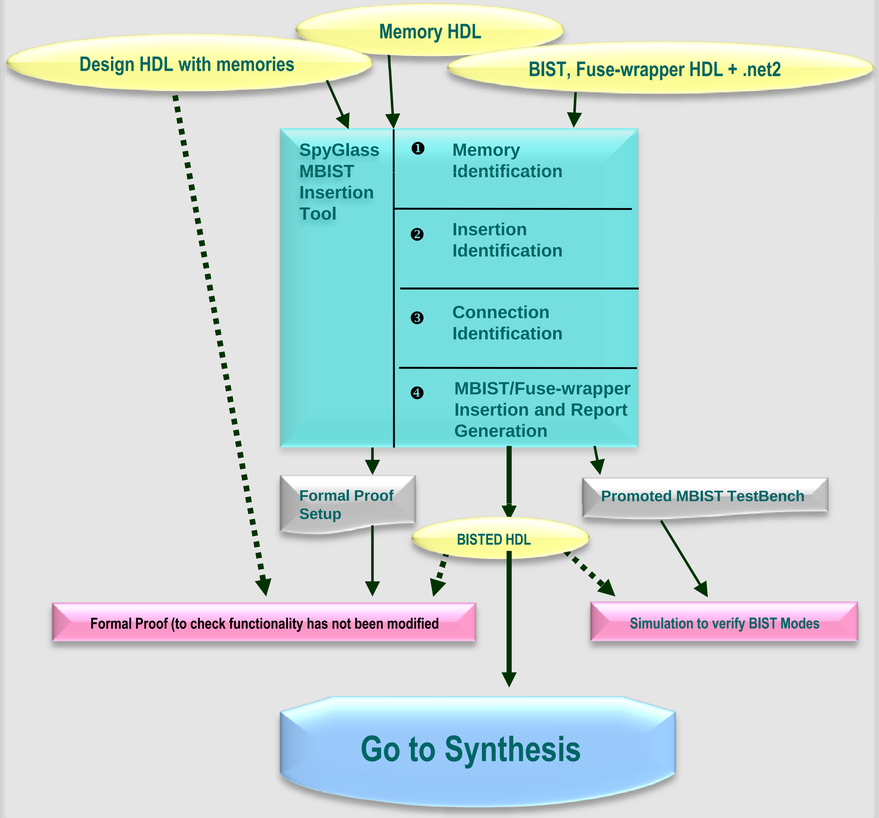
Design Related Issues

- Module from the customer
- Many Memory Cuts
- Risk of Memory Change
- Memories running at high speed



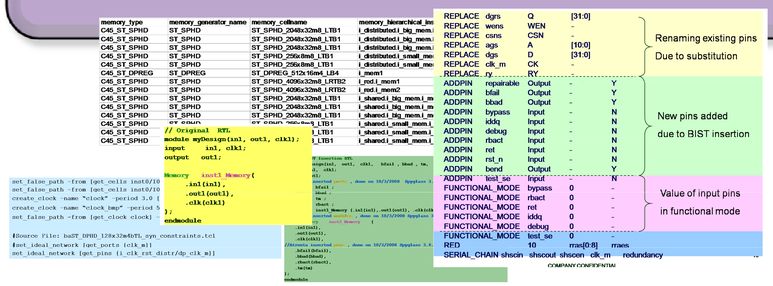
- BIST insertion transparent for RTL designers
- Fast and safe response to memory changes
- BIST insertion at RTL level

MBIST Insertion Flow with Atrenta SpyGlass-MBIST



Details of the Implemented Flow

- Dedicated BIST for each Memory (Sharing BIST is also possible in the tool)
- BIST insertion done at RTL level with some gate level modules
- Top level insertion (Hierarchical flow also supported)
- Manual connection of the Fuse and repair logic (Tool supports automation for Fuse and repair logic connection)



memory_type	memory_generator_name	memory_cellname	memory_hierarchical_be
C45_ST_SPHD	ST_SPHD	ST_SPHD_2048x32m8_LTB1	i_dsbtituted.i_blg_mem.i
C45_ST_SPHD	ST_SPHD	ST_SPHD_2048x32m8_LTB1	i_dsbtituted.i_blg_mem.i
C45_ST_SPHD	ST_SPHD	ST_SPHD_2048x32m8_LTB1	i_dsbtituted.i_blg_mem.i
C45_ST_SPHD	ST_SPHD	ST_SPHD_256x8m8_LTB1	i_dsbtituted.i_small_mem
C45_ST_SPHD	ST_SPHD	ST_SPHD_256x8m8_LTB1	i_dsbtituted.i_small_mem
C45_ST_DPREG	ST_DPREG	ST_DPREG_512x16m4_LB4	i_mem1
C45_ST_SPHD	ST_SPHD	ST_SPHD_4096x32m8_LRTB2	i_md.i_mem1
C45_ST_SPHD	ST_SPHD	ST_SPHD_4096x32m8_LRTB2	i_md.i_mem2
C45_ST_SPHD	ST_SPHD	ST_SPHD_2048x32m8_LTB1	i_shared.i_blg_mem.i_m
C45_ST_SPHD	ST_SPHD	ST_SPHD_2048x32m8_LTB1	i_shared.i_blg_mem.i_m
C45_ST_SPHD	ST_SPHD	ST_SPHD_2048x32m8_LTB1	i_shared.i_blg_mem.i_m
C45_ST_SPHD	ST_SPHD	ST_SPHD_2048x32m8_LTB1	i_shared.i_small_mem.i
C45_ST_SPHD	// Original RTL		i_shared.i_small_mem.i
C45_ST_SPHD	module myDesign(in1, out1, clk1);	LTB1	i_shared.i_small_mem.i

```
input    in1, clk1;
output   out1;

Memory   inst1 Memory(
    .in1(in1),
    .out1(out1),
    .clk(clk1)
);
endmodule
```

```
set_false_path -from [get_cells inst0/10
set_false_path -from [get_cells inst0/10
create_clock -name "clock" -period 3.0 {
create_clock -name "clock_bmp" -period 5
set_false_path -from [get_clock clock] -
```

```
#Source File: ba3T_DPHD_128x32m8bTL_syn_constraints.tcl
#set_ideal_network [get_ports {clk_m}]
set_ideal_network [get_pins {i_clk_rst_distr/dp_clk_m}]
```

REPLACE	dgrs	Q	[31:0]		Renaming existing pins Due to substitution
REPLACE	wons	WEN	-		
REPLACE	cons	CSN	-		
REPLACE	ags	A	[10:0]		
REPLACE	dgs	D	[31:0]		
REPLACE	clk_m	CK	-		
REPLACE	ry	RY			
ADDPIN	repairable	Output	-	Y	New pins added due to BIST insertion
ADDPIN	bfail	Output	-	Y	
ADDPIN	bfaid	Output	-	Y	
ADDPIN	bypass	Input	-	N	
ADDPIN	iddq	Input	-	N	
ADDPIN	debug	Input	-	N	
ADDPIN	rbact	Input	-	N	
ADDPIN	ret	Input	-	N	
ADDPIN	rst_n	Input	-	N	
ADDPIN	bend	Output	-	Y	
ADDPIN	test_se	Input	-	N	Value of input pins in functional mode
FUNCTIONAL_MODE	bypass	0	-		
FUNCTIONAL_MODE	rbact	0	-		
FUNCTIONAL_MODE	ret	0	-		
FUNCTIONAL_MODE	iddq	0	-		
FUNCTIONAL_MODE	debug	0	-		
FUNCTIONAL_MODE	test_se	0			
RFID		10	rma[0:8]	rmaws	
SERIAL_CHAIN shscin		shscout shscen	clk_m	redundancy	

COMPANY CONFIDENTIAL

Summary and Results

Benefits:

- Fast, predictable and reproducible flow.
  - 8h runtime with the gate level submodules
  - <1h runtime if just the RTL modules

  on a 2.5M gate design with 92 memory cuts
- Flexible flow: RTL flow used with some gate level modules
- Easy change of memories and different MBIST strategies
- Timing-optimization of complete RTL, including BIST, during synthesis
- Simulation of MBIST performed both at RTL and gate

Marcello Raimondi marcello.raimondi@st.com Alberto Caravà alberto.carava@st.com Frederic Grandvaux frederic.grandvaux@st.com