# Generating Scan Patterns for Scan Chains with Intermittent Timing Problems

Richard Illman
Dialog Semiconductor

June 25th, 2015
SNUG Munich

# Agenda

Scan Chain Design Issues

Problems of Diagnosing Failures

New Approach

Example Design

Production Patterns for Designs with Timing Issues

Conclusions

# Scan Chain Design Issues

- Flop-to-flop paths are short
  - Q output to SI input
  - Clock skew between different domains may be significant
  - High peak switching during scan shift may cause IR drop issues

- lockup latches prevent race hazards at domain boundaries
  - DC
    - `set_scan_configuration -internal_clocks multi`
    - `set_scan_configuration -add_lockup true`
  - TetraMAX
    - `report_lockup_latches -gated_clock \`
      `-latch_based_balanced`

# Scan Chain Continuity Patterns

```
set_atpg -chain_test \
      <off | 0011 | 0101 | 1000 | 0111 | string>

string
011001R --> 011001 011001 011001  R=repeat sequence
011011C --> 011011 111111 111111  C=repeat last charecter
```

- Specialised patterns can be created
  - e.g. Low switching activity

# Diagnosing Chain Failure

Synopsys Users Group

Diagnosing the failure type is simple

Diagnosing the location is difficult

Fail signature is the same no matter where the fail is located

00110011 →  → LLHHLLHH

```
LLHHLLHH   Expected
.LLHHLLH   Delay (one cycle)
LHHLLHH.   Race (one cycle)
LHHHLHHH   Race (data rising)
LLLHLLLH   Race (data falling)
LLLLLLLL   ???
```

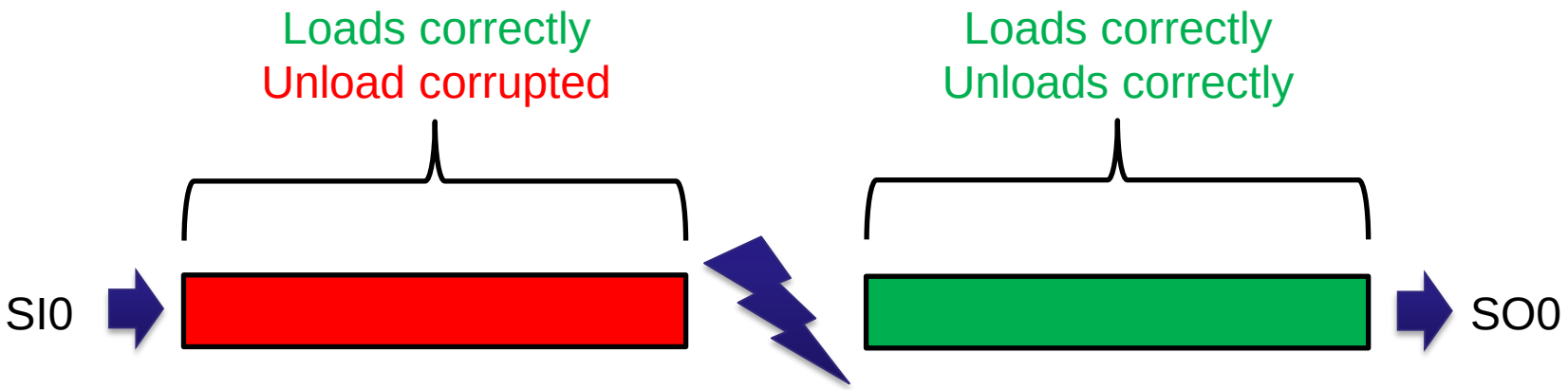
# Set/Reset Test

```
set_atpg -add_setreset_test
```

- Load chain with 0s
- Pulse set/reset
- Unload chain

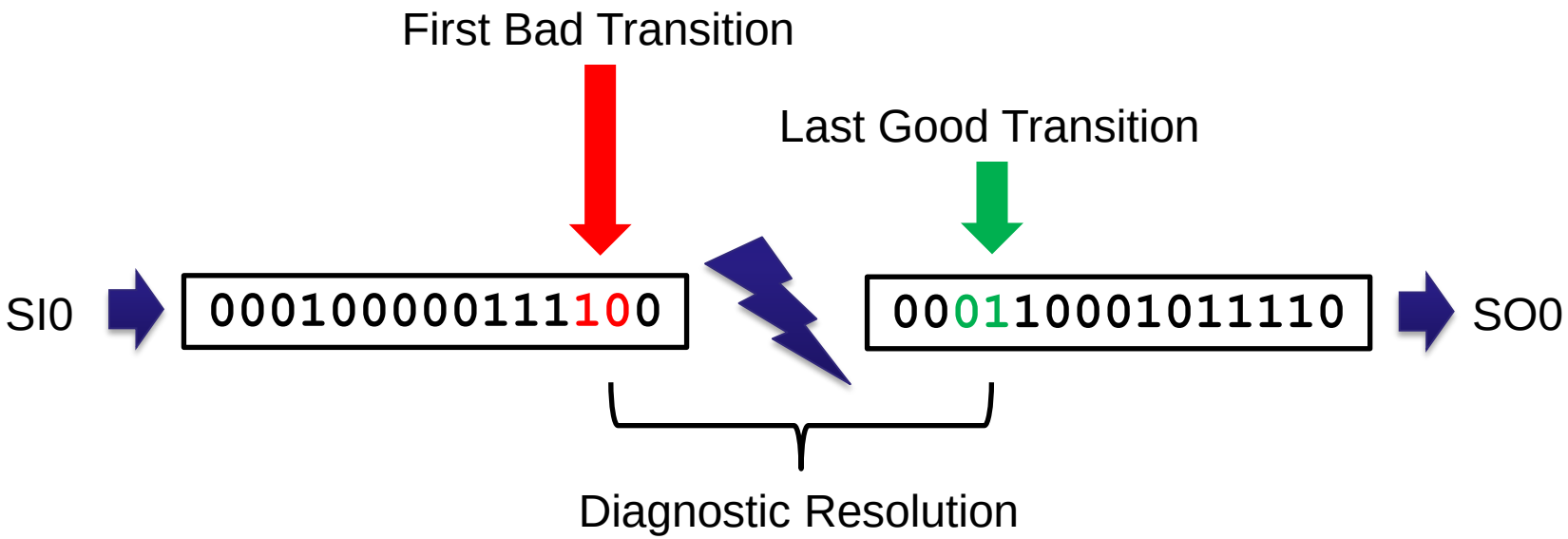
- Provides information on fail location

# Set/reset test




- Load of a constant value pattern is unaffected by race condition
- The set/reset pulse creates a different pattern
- During unload data transitions between the scan-in and race location are corrupted
  - Location of the timing issue can be localised

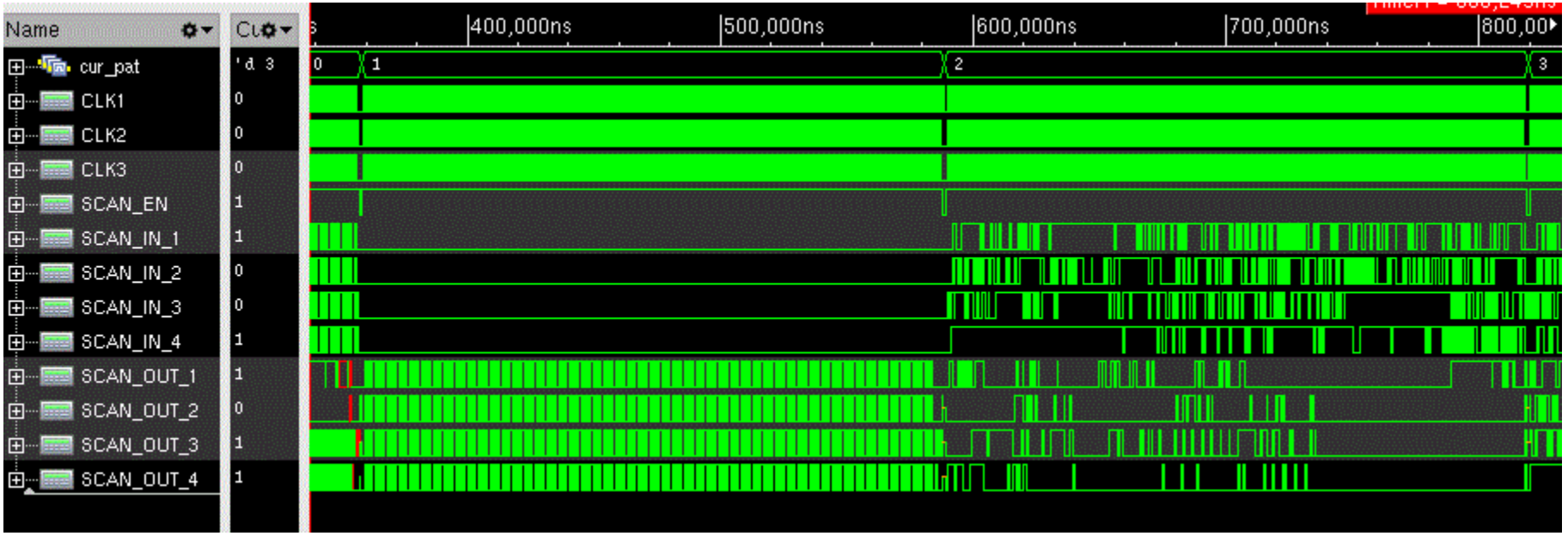
# Set/reset test



# Set/reset test

Pulse set/reset

Load 0s




Unload set/reset values

# Set/reset test improvements

- Modify/Add test to load constant 1 instead of constant 0
- Use other capture clock sequences

# Set/reset test limitations

- Limited possible patterns
  - Load zero, pulse set/reset
    - Non-set/reset flops=0, set flops=1, reset flops=0
  - Load one, pulse set/reset
    - Non-set/reset flops=1, set flops=1, reset flops=0
- Often flops are mostly reset
  - Very few transitions in the unload data
  - Little localisation
- Design Options
  - Counter & FSM start values of hex 55 or AA
  - Reorder chains to mix flop types
  - Unpopular, costly, hopefully not needed

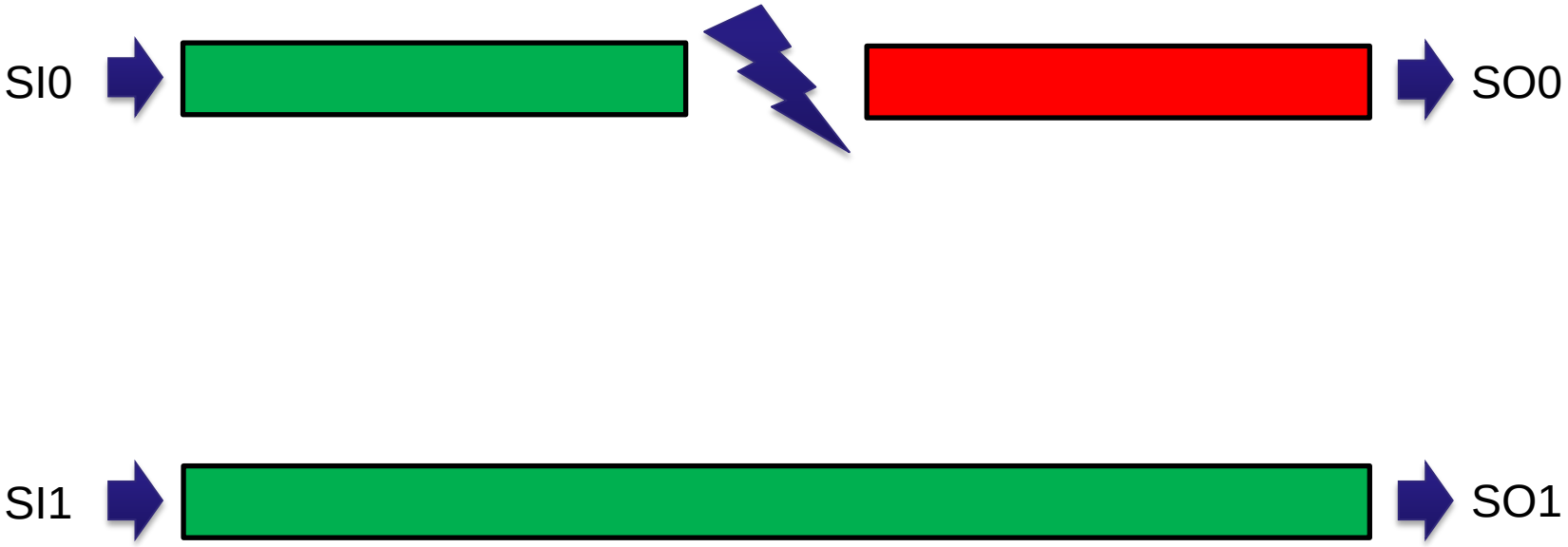
# Scan chain diagnostics

- Location of scan chain failures can be diagnosed within TetraMAX

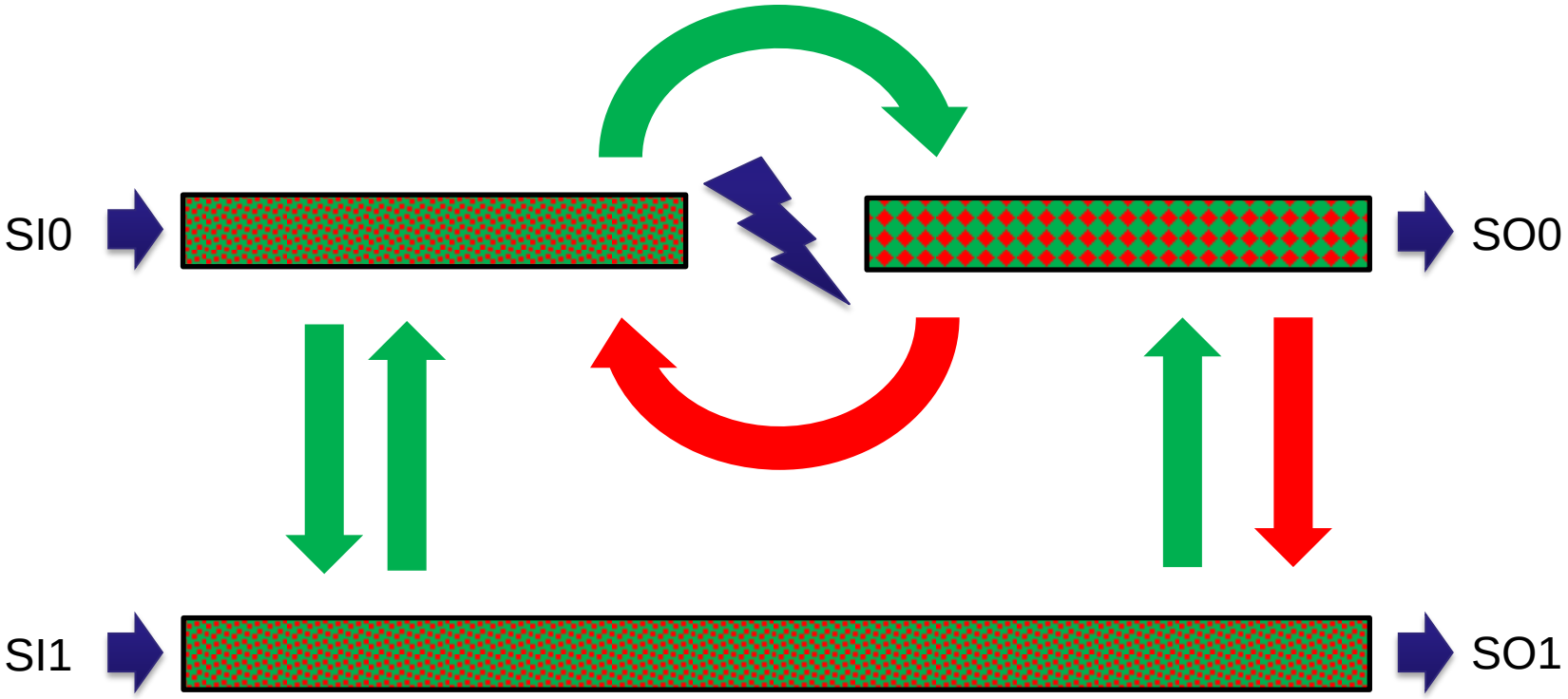
- Simulated effects of different fail modes/locations are “scored” against fails observed on the tester.

- Issues
  - Large volumes of fail data required
    - Fail memory may be limited on the tester
  - Location may not be accurately diagnosed
    - More difficult than for fails outside the scan chains
    - Intermittent failures affect accuracy

# Load



# Capture/Unload



# Advantages of the set/reset test

- Advantages
  - **The test uses load patterns unaffected by the timing issue**
  - Fails are seen only in the faulty chain
  - Fails easily diagnosed and  located between the last good transition and the first bad transition

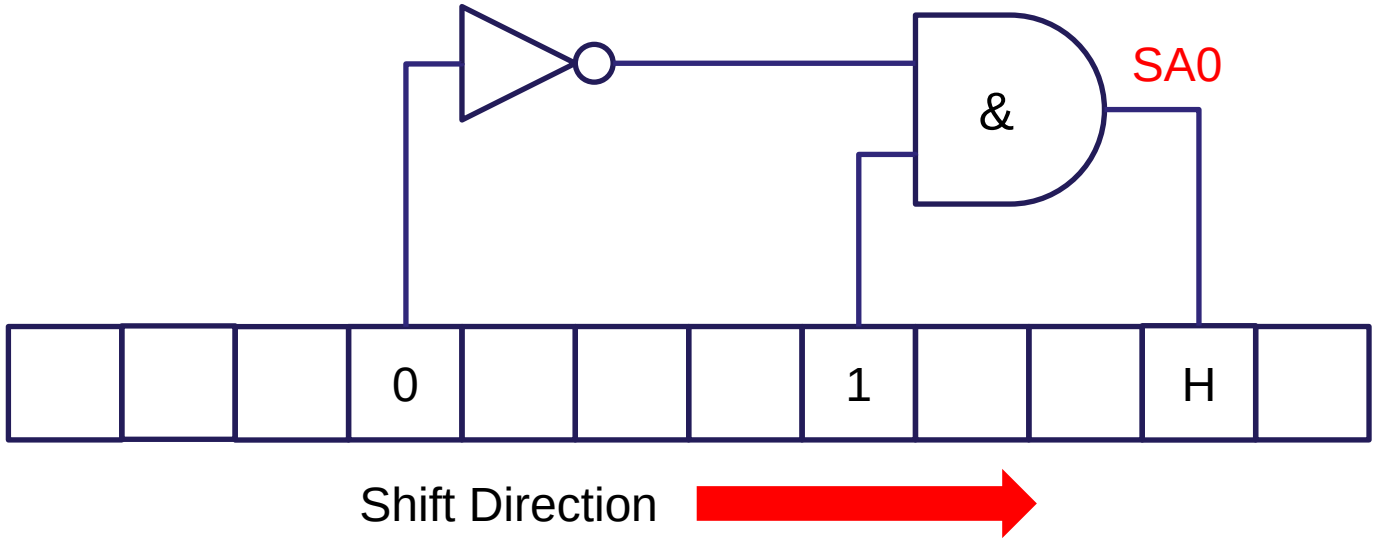
- Disadvantages
  - Only two different possible load patterns, all-0, all-1
  - Only two possible unload patterns

# Requirements for improved scan chain diagnostics

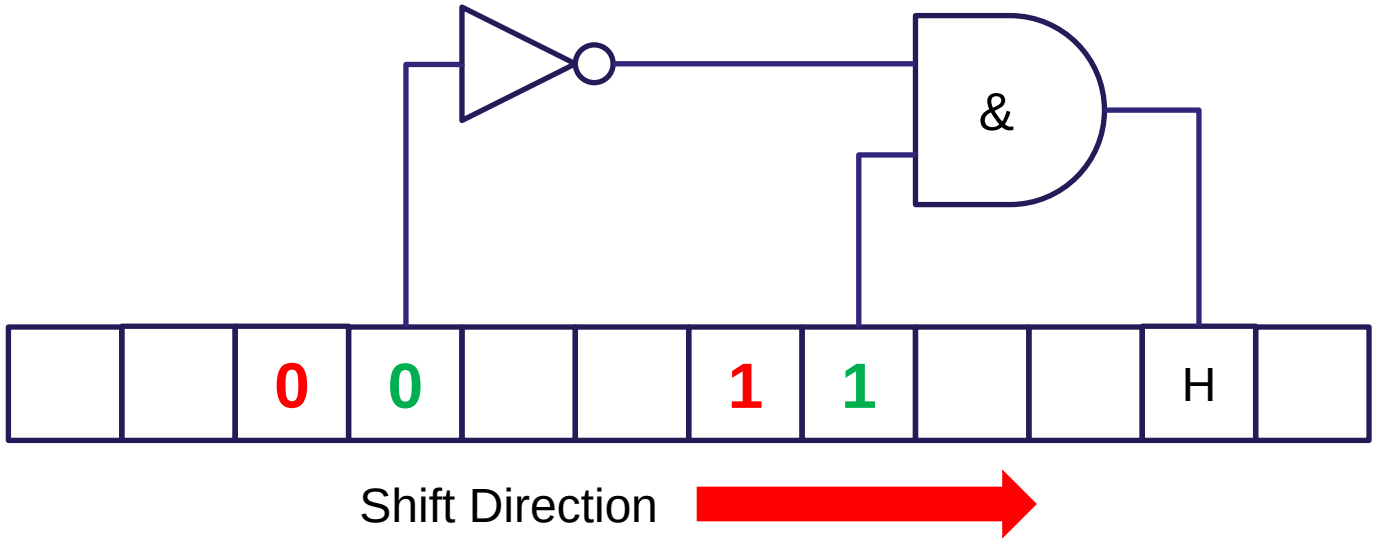
- Load patterns are unaffected by the timing issue
- Different unload patterns giving transitions across more of the chain
- ***Apparently*** there are only two suitable load patterns
  - All-zero and All-one

# Example ATPG Pattern

Testing for SA-0 on AND gate inputs/output



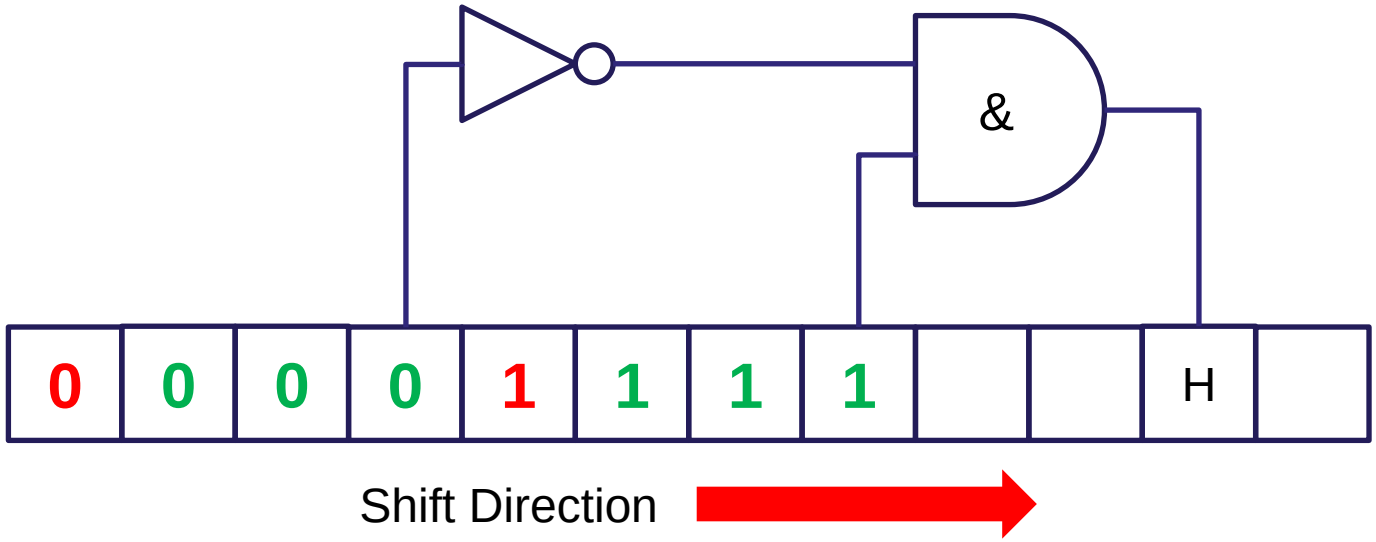
# Use of “Guard Values”




“Guard values” ensure that the gate “sees” the correct input values in the presence (or absence) of a race condition anywhere in the scan chain.

The guard values themselves must be simulated as “X” but physically present in the test data applied to the device

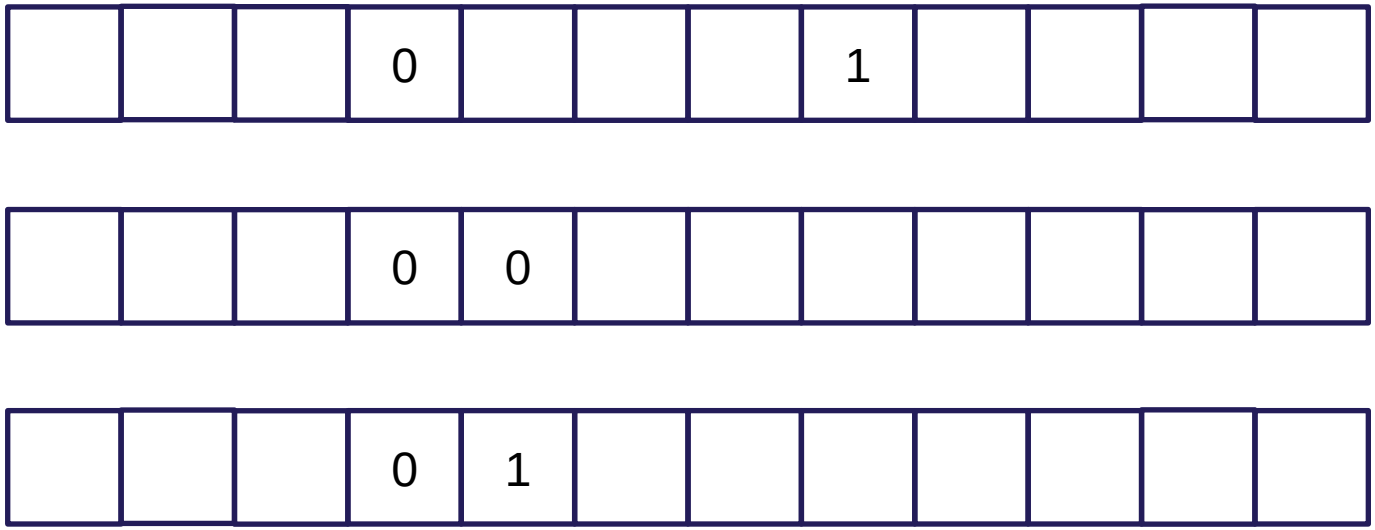
# Repeat Values




Repeat values can be used to minimise undefined values.
Guard bits are only needed between conflicting “care” values

# ATPG Requirements

Generate patterns with no adjacent and conflicting care bits in the **load** pattern for the faulty scan chain



# Generating Patterns – Stage 1

```
set_atpg -nouse_conflicting_adjacent_load_care_bits \
         -chain chain_name
```

```
set_atpg -fill X;          # allow only the care bits to be seen

set_atpg -merge off;       # prevent conflicts from different tests

set_atpg -decision random;
run_atpg -ndetects d;      # generate different patterns
```

Do not use "`run_atpg -optimize`" or "`run_atpg basic -auto`"
ATPG setting are ignored/modified

# Stage 1

```
XXXX1XXXX0XXXXXXX1XX Load
XXXXXXXXHXXXXXXXLHXXX Unload
```

Initial ATPG

# Generating Patterns – Stage 2

- Modify the STIL file with external scripts
  - Add the “repeat” bits into the faulty chain(s)
    - The guard bits must remain as X
  - Add fill (random) to any other chains
  - Change all expect bits in the unload data to zero
- `run_simulation -override_differences`
  - # Change the “dummy” expect values to logically correct ones

# Stage 2

```
XXXX1XXXX0XXXXXX1XX Load
XXXXXXXXHXXXXXXLHXXX Unload
```

Initial ATPG

```
X1111X0000X111111XX Load
LLLLLLLLHLLLLLLLLHLLL Unload
```

Add repeat bits & fill the expect bits

```
X1111X0000X111111XX Load
LLHHXXXHXLXHXXLHXLL Unload
```

```
run_simulation
-override_differences
```

# Generating Patterns – Stage 3

- Fault simulate the patterns to minimise the fault set
  - `run_simulation -detected_pattern_storage`
  - `write_faults filename -class DT`
- Parse the fault file to find the effective pattern numbers and create a reorder file
- Write only the effective patterns as a STIL file
  - `write_patterns filename -reorder file -format stil`
- Process the STIL file to add the guard bits into the STIL

# Stage 3

```
XXXX1XXXX0XXXXXX1XX Load
XXXXXXXXHXXXXXXLHXXX Unload
```

Initial ATPG

```
X1111X0000X111111XX Load
LLLLLLLLHLLLLLLLLHLLL Unload
```

Add repeat bits & fill the expect bits

```
X1111X0000X111111XX Load
LLHHXXXHXLXHXXLHXLL Unload
```

```
run_simulation
-override_differences
```

```
1111100000111111XX Load
LLHHXXXHXLXHXXLHXLL Unload
```

Add guard bits

# Generating Patterns – Stage 4

- Convert patterns to tester format and run/log as standard

- Diagnosing failures
  - Find the lower failing bit number in the unload
  - Run conventional TMAX chain diagnostics

```
run_diagnosis file_name \
   [-assume_chain_defect fast chain_name >]
```

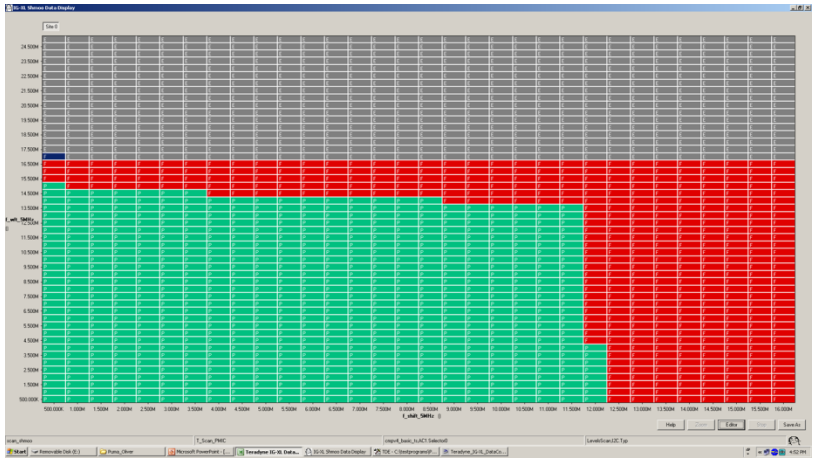
# Example Design

- Counter block with around 180 scan flops.
  - Two scan sub-chains of 90 bits each.
  - Separate chains in compression mode
  - Contained within 2 longer chains in full (legacy) scan mode
  - Contained in analogue block outside main digital
- Both chains showed intermittent race conditions on some devices (about 20%).
- Standard diagnostics did not identify the location

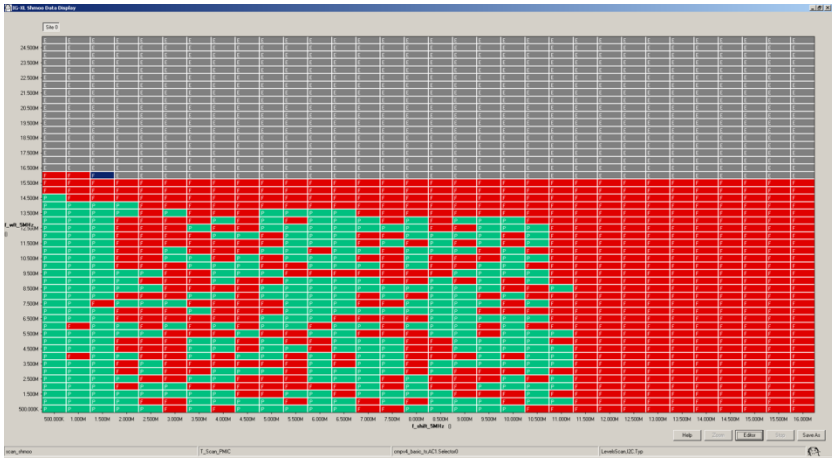
# Example Design – shmoo plots

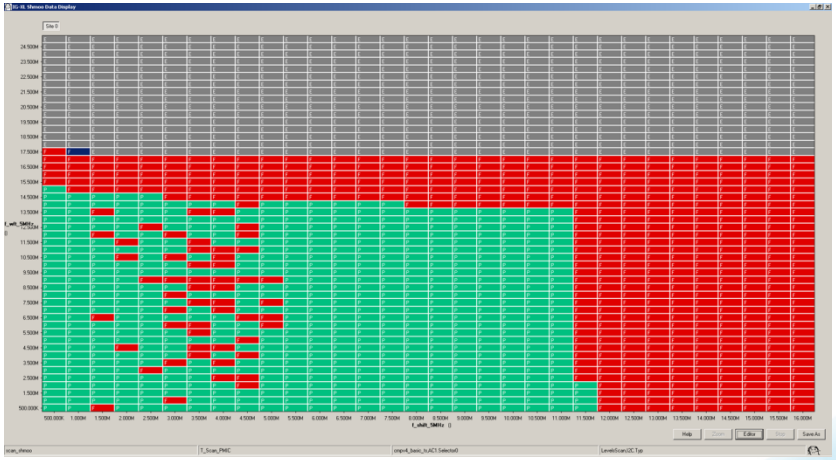
Synopsys Users Group

## Good Device



## Example Faulty Devices





# Example Design

- New diagnostic approach
  - First chain diagnosed to failure between bits 20 and 0 (scan out)
  - Second chain diagnosed to failure between bits 8 and 0.

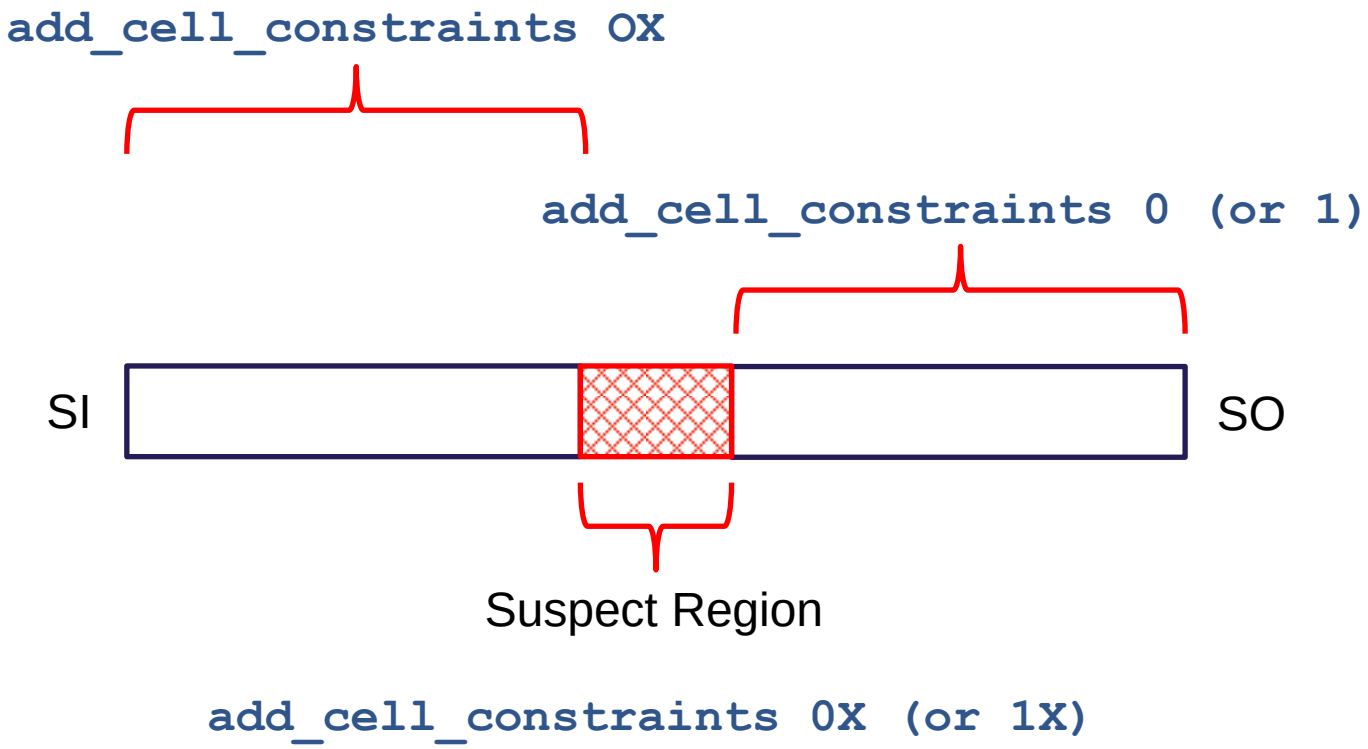
# Production Patterns

- Hierarchy of possible approaches

- **`add_cell_constraints XX` *`chain_name`***

- **`add_cell_constraints 0X` *`chain_name`***
- **`add_cell_constraints 1X` *`chain_name`***

# Partially diagnosed chain




If the “suspect region” is close to the scan-out pin there is very little coverage

# New Technique

- Generate patterns as for diagnostics
  - Use of guard bits in the load data
- Look for “unloaded guarded” bits in the unload data
  - The following bit is the same
  - Replace the “unload guard bits” with X

  - **Original Unload** **LLXHHHXXLXHHHXLL**
  - **Modified Unload** **XLXXHHXXXXXHHXXL**

- Fault simulate the patterns to check coverage
  - Value will be the coverage in the error free circuit
  - Functionally we aren’t concerned with scan chain race conditions

# Pattern Verification

- Generate the testbench in “serial only mode”
  - `stil2verilog -ser_only`
  - Independent of internal netlist structure
- Generate modified netlist(s) with “race conditions”
  - Bypass one of the flops in the chain
  - Typically try the first/last bit positions
- Simulate the testbench and check that there are no failures

# Results – Example Design

- Two chains both had errors near the scan-out pins
- OX (observe X) constraint on the full chain
- Isolated from main digital block
- The modified continuity pattern gave ~20% coverage

- New patterns gave coverage ~80%

# Issues

- These techniques only work on uncompressed (legacy) scan
  - ATPG “fill” options not supported for compression
  - Failing chain diagnosis from compression mode can narrow the target are for initial diagnosis
- Increased pattern counts due to “no merge ATPG”
  - Not an issue for diagnostics
- Large intermediate STIL files during the generation process

# Conclusions

- The quality of scan chain diagnostics can be improved by use of load patterns which are unaffected by the timing issue.

- It is possible to create limited scan patterns which work reliably even in the presence of scan chain timing issues.

- New features for DFTMAX Ultra
  - `add_chain_masks`

# Thank You

# Selecting the fault list

- Faults on the D inputs of flops in the faulty chain

- Add “OX” cell constraints and “capture masks” on flops in all other chains
  - Only faults feeding flops in the faulty chain will be testable

# Production Patterns

- Create a timing tolerant scan continuity pattern
  - `set_atpg -chain_test 00001111R`
- Modify the expect values in the STIL file
  - **LLLLHHHHLLLLHHHH**
  - **XLLXXHHXXLLXXHHX**
- Provides around 10~20% coverage
  - Checks flops/clock are functional