Overview of Formality Labs for Debugging Failing Verifications

Purpose: These labs are designed for you to find, analyze, and solve common equivalency checking problems using Formality. You can use these labs to increase your awareness of Formality and to practice debugging skills.

Content: These labs use public-domain RTL source. The netlists were generated using Design Compiler F-2011.09 release software.

Procedure:

- There is a README file for every lab describing what to do.
- Each lab has a "runme.fms" FM Tcl script you can use initially.
- Each lab has a "hint" directory containing a README file if you need some helpful pointers on what to do.
- If you find that a lab is too difficult, there is a ".solution" sub-directory with the correct solution.
- Please compare your results with the correct results when you finish each lab.
- This lab document will guide you through each lab.

Invoke Formality in this manner:

"fm_shell -gui -f runme.fms |tee runme.log" or

"formality -f runme.fms |tee runme.log"

FM Lab1: Missing Verification Files

Objective: This lab shows an example of what happens in verification if pieces of the reference and implementation designs are missing and if guidance is missing. The focus of this lab is to review transcript messages. You need to change the "runme.fms" FM Tcl script to get a successful verification.

Lab flow:

1.) Run the verification using the existing "runme.fms" script.

2.) Finding clues to indicate potential problems:

2a) Transcript messages:

Formality debugging involves collecting information that may point to the reason why the design fails verification. Always look at the transcript messages first.

Note the following warning messages in the transcript:

```
Status:  Creating black-box designs...
Created technology library 'FM_BBOX' in container 'r' for black-box designs
Created black-box design 'mAlu' in library 'FM_BBOX'
Warning: 1 blackbox designs were created for missing references. (FM-064)
Status:  Attempting to resolve unlinked cells by using black-boxes...
Warning: 150 black-box pins of unknown direction found; see formality.log for list
(FM-230)
Top design set to 'r:/WORK/mR4000' with warnings
```

Formality is creating a black-box in the reference design to represent a missing piece of the design. The missing piece is “mAlu”. Perhaps an engineer forgot to send over that portion of the RTL, or merely left it out of the FM Tcl script.

This transcript message is only a warning instead of an error because the customer included this variable setting in the FM TCL script:

```
set hdlin_unresolved_modules black_box
```

Otherwise, by default Formality would have stopped processing the design.

When faced with a missing piece of the reference design, you can either find the missing piece and try verification again. Or, you can try to black-box the equivalent sub-design in the implementation design, if the hierarchy was retained during synthesis.

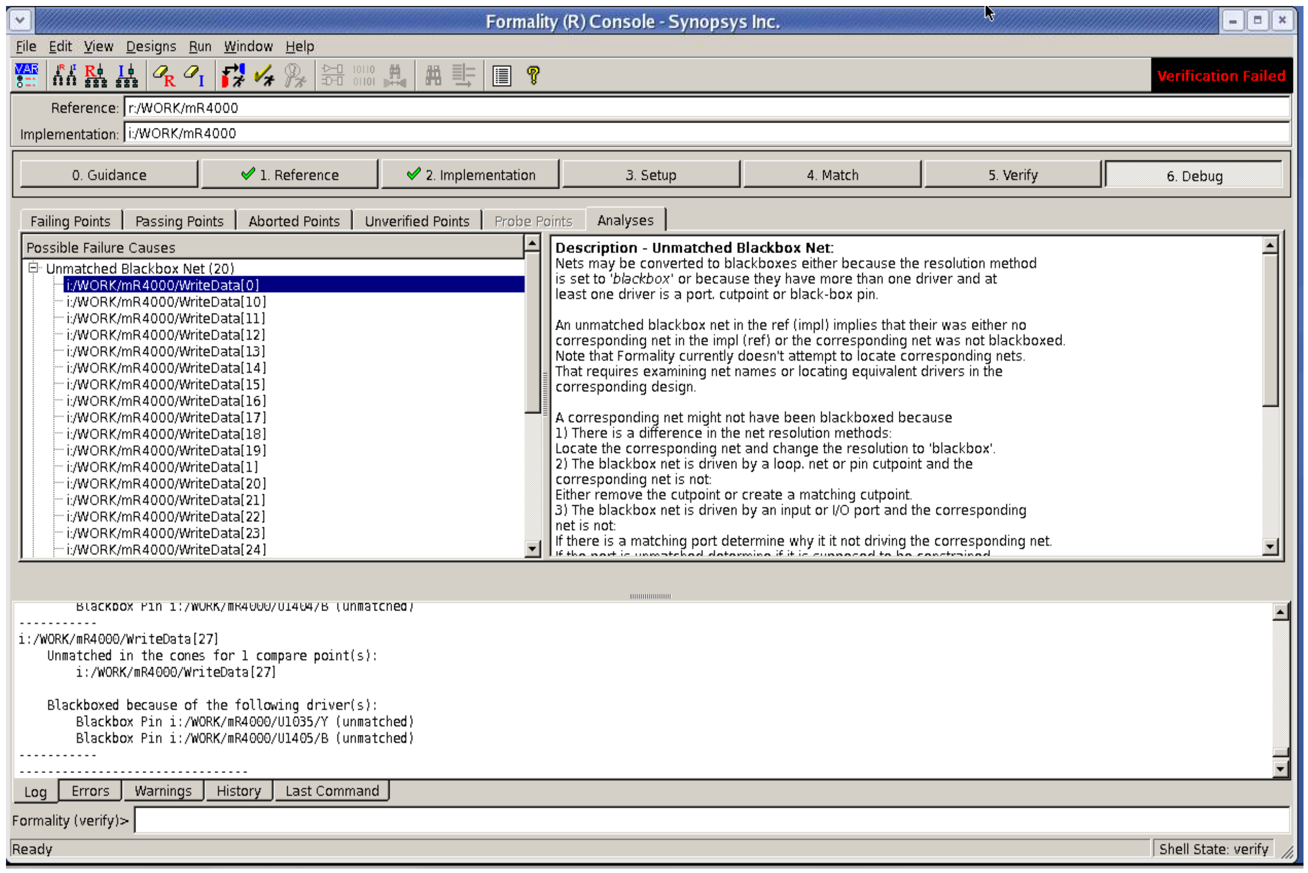
3.) For this lab, you can find the missing RTL by quickly browsing in the “rtl” sub-directory and include the missing file in your FM Tcl script. Re-run verification.

4.) After running verification again, notice that the transcript still has these messages:

```
set_top i:/WORK/mR4000
Setting top design to 'i:/WORK/mR4000'
Warning: Cannot link cell '/WORK/mR4000/DP_OP_94J1_124_8045_U32' to its reference
design 'fa2a0'. (FE-LINK-2)
Warning: Cannot link cell '/WORK/mR4000/DP_OP_94J1_124_8045_U31' to its reference
design 'fa1b0'. (FE-LINK-2)
Warning: Cannot link cell '/WORK/mR4000/DP_OP_94J1_124_8045_U30' to its reference
design 'fa2a0'. (FE-LINK-2)
Warning: Cannot link cell '/WORK/mR4000/DP_OP_94J1_124_8045_U29' to its reference
design 'fa1b0'. (FE-LINK-2)
Warning: Cannot link cell '/WORK/mR4000/DP_OP_94J1_124_8045_U28' to its reference
design 'fa2a0'. (FE-LINK-2)
Warning: Cannot link cell '/WORK/mR4000/DP_OP_94J1_124_8045_U27' to its reference
design 'fa1b0'. (FE-LINK-2)
Warning: Cannot link cell '/WORK/mR4000/DP_OP_94J1_124_8045_U26' to its reference
design 'fa2a0'. (FE-LINK-2)
Warning: Cannot link cell '/WORK/mR4000/DP_OP_94J1_124_8045_U25' to its reference
design 'fa1b0'. (FE-LINK-2)
```

This indicates that Formality cannot find several library component cells for the implementation design. Formality is creating black-boxes for them. This is a sign that a library is missing from the setup.

5.) Since verification already ran, if you run the "Analyze" command, Formality will indicate that there are unmatched black-box nets in the implementation design that do not exist in the reference design. This is another indication of something missing in the implementation design.



```
Unmatched Blackbox Net (20)
  i:/WORK/mR4000/WriteData[0]
  i:/WORK/mR4000/WriteData[10]
  i:/WORK/mR4000/WriteData[11]
  i:/WORK/mR4000/WriteData[12]
  i:/WORK/mR4000/WriteData[13]
  i:/WORK/mR4000/WriteData[14]
  i:/WORK/mR4000/WriteData[15]
  i:/WORK/mR4000/WriteData[16]
  i:/WORK/mR4000/WriteData[17]
  i:/WORK/mR4000/WriteData[18]
  i:/WORK/mR4000/WriteData[19]
  i:/WORK/mR4000/WriteData[1]
  i:/WORK/mR4000/WriteData[20]
  i:/WORK/mR4000/WriteData[21]
  i:/WORK/mR4000/WriteData[22]
  i:/WORK/mR4000/WriteData[23]
  i:/WORK/mR4000/WriteData[24]
```

Description - Unmatched Blackbox Net:
Nets may be converted to blackboxes either because the resolution method is set to '*blackbox*' or because they have more than one driver and at least one driver is a port. cutpoint or black-box pin.

An unmatched blackbox net in the ref (impl) implies that their was either no corresponding net in the impl (ref) or the corresponding net was not blackboxed. Note that Formality currently doesn't attempt to locate corresponding nets. That requires examining net names or locating equivalent drivers in the corresponding design.

A corresponding net might not have been blackboxed because
1) There is a difference in the net resolution methods:
Locate the corresponding net and change the resolution to 'blackbox'.
2) The blackbox net is driven by a loop. net or pin cutpoint and the corresponding net is not:
Either remove the cutpoint or create a matching cutpoint.
3) The blackbox net is driven by an input or I/O port and the corresponding net is not:
If there is a matching port determine why it it not driving the corresponding net.

```
        Blackbox Pin i:/WORK/mR4000/U1404/B (unmatched)
-----------
i:/WORK/mR4000/WriteData[27]
    Unmatched in the cones for 1 compare point(s):
        i:/WORK/mR4000/WriteData[27]

    Blackboxed because of the following driver(s):
        Blackbox Pin i:/WORK/mR4000/U1035/Y (unmatched)
        Blackbox Pin i:/WORK/mR4000/U1405/B (unmatched)
-----------
--------------------------------
```

6.) Find the missing library and include it in the FM Tcl script. Re-run verification.

7.) During this verification run, Formality located all of the design pieces and library information. However, verification is still failing. The only clue for this issue is the following statement in the transcript:

```
Info:  Formality Guide Files (SVF) can improve verification success by automating
setup.
```

You need to include the SVF guidance file in the FM Tcl script:

```
set_svf mR4000.svf
```

8.) Since there is no clock-gating nor scan involved, there is no additional setup needed. Auto setup mode is not required.

9.) Try verification again. You should now get a successful verification.

9.) You can automatically create a FM Tcl script by using UNIX command "fm_mk_script". Try the following:

```
fm_mk_script mR4000.svf
```

10.)View the resulting FM Tcl script "fm_mk_script.tcl", and try it out with Formality.

FM Lab2: Synthesis Pragmas

Objective: This lab contains Verilog RTL using Synopsys Parallel Case and Full Case synthesis pragmas. You must change the "runme.fms" FM TCL script to get a successful verification.

Lab flow:

1.) Run the verification using the existing "runme.fms" script.

2.) Find clues to indicate potential problems.

2a) Transcript messages:

Formality debugging involves collecting information that may point to the reason why the design fails verification. Always review the transcript messages first.

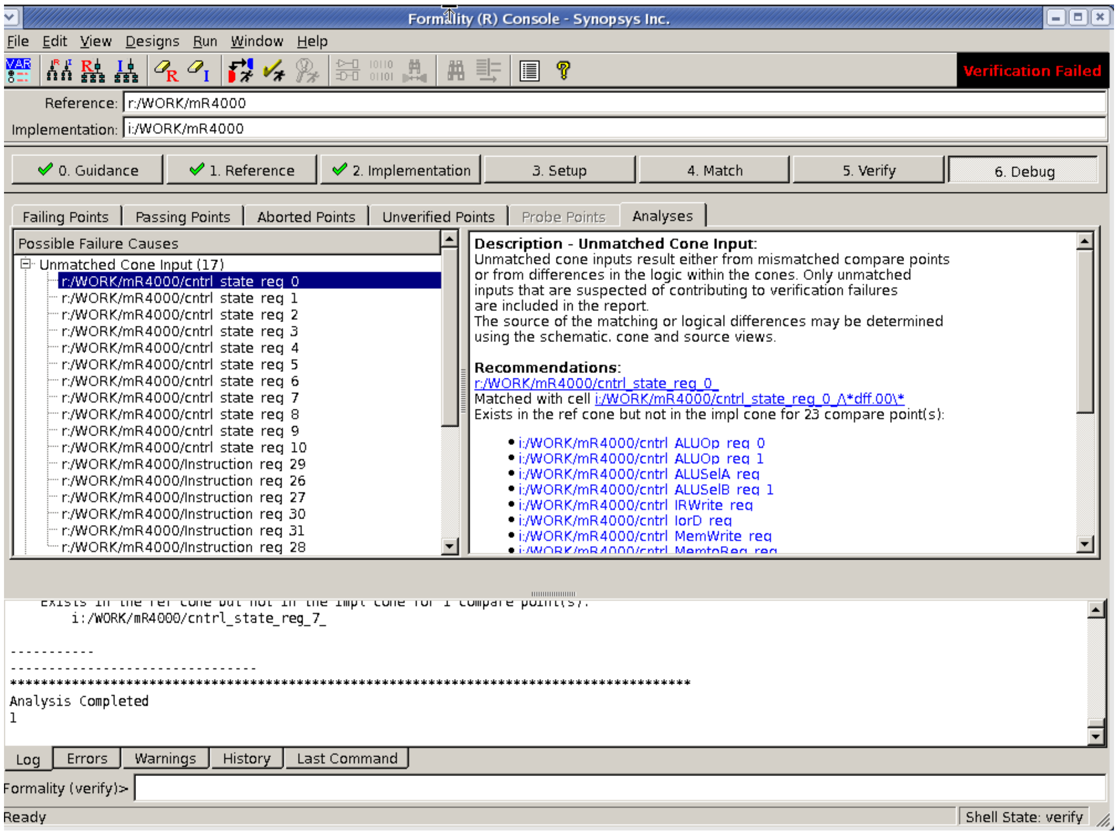
Note the following warning message in the transcript:

```
************ RTL Interpretation Summary ************
************ Design: r:/WORK/mR4000
full_case ignored (7 total, 1 with unspecified cases)
parallel_case ignored (7 total, 1 with overlapping cases)

Please refer to the Formality log file for more details,
or execute report_hdlin_mismatches.
****************************************************
```

This is our first clue that there may be simulation/synthesis mismatches due to specifying full case and parallel case pragmas in the RTL.

2b) Messages from analyze_point commands:

Under the Debug tab, run "Analyze". Formality knows that the logic cones are different, but cannot pinpoint the specific problem.



Reference: r:/WORK/mR4000
Implementation: i:/WORK/mR4000

Possible Failure Causes

- Unmatched Cone Input (17)
  - r:/WORK/mR4000/cntrl state reg 0
  - r:/WORK/mR4000/cntrl state reg 1
  - r:/WORK/mR4000/cntrl state reg 2
  - r:/WORK/mR4000/cntrl state reg 3
  - r:/WORK/mR4000/cntrl state reg 4
  - r:/WORK/mR4000/cntrl state reg 5
  - r:/WORK/mR4000/cntrl state reg 6
  - r:/WORK/mR4000/cntrl state reg 7
  - r:/WORK/mR4000/cntrl state reg 8
  - r:/WORK/mR4000/cntrl state reg 9
  - r:/WORK/mR4000/cntrl state reg 10
  - r:/WORK/mR4000/Instruction reg 29
  - r:/WORK/mR4000/Instruction reg 26
  - r:/WORK/mR4000/Instruction reg 27
  - r:/WORK/mR4000/Instruction reg 30
  - r:/WORK/mR4000/Instruction reg 31
  - r:/WORK/mR4000/Instruction reg 28

Description - Unmatched Cone Input:
Unmatched cone inputs result either from mismatched compare points
or from differences in the logic within the cones. Only unmatched
inputs that are suspected of contributing to verification failures
are included in the report.
The source of the matching or logical differences may be determined
using the schematic. cone and source views.

Recommendations:
r:/WORK/mR4000/cntrl_state_reg_0_
Matched with cell i:/WORK/mR4000/cntrl_state_reg_0_/*dff.00*
Exists in the ref cone but not in the impl cone for 23 compare point(s):

- i:/WORK/mR4000/cntrl ALUOp reg 0
- i:/WORK/mR4000/cntrl ALUOp reg 1
- i:/WORK/mR4000/cntrl ALUSelA reg
- i:/WORK/mR4000/cntrl ALUSelB reg 1
- i:/WORK/mR4000/cntrl IRWrite reg
- i:/WORK/mR4000/cntrl IorD reg
- i:/WORK/mR4000/cntrl MemWrite reg
- i:/WORK/mR4000/cntrl MemtoReg reg

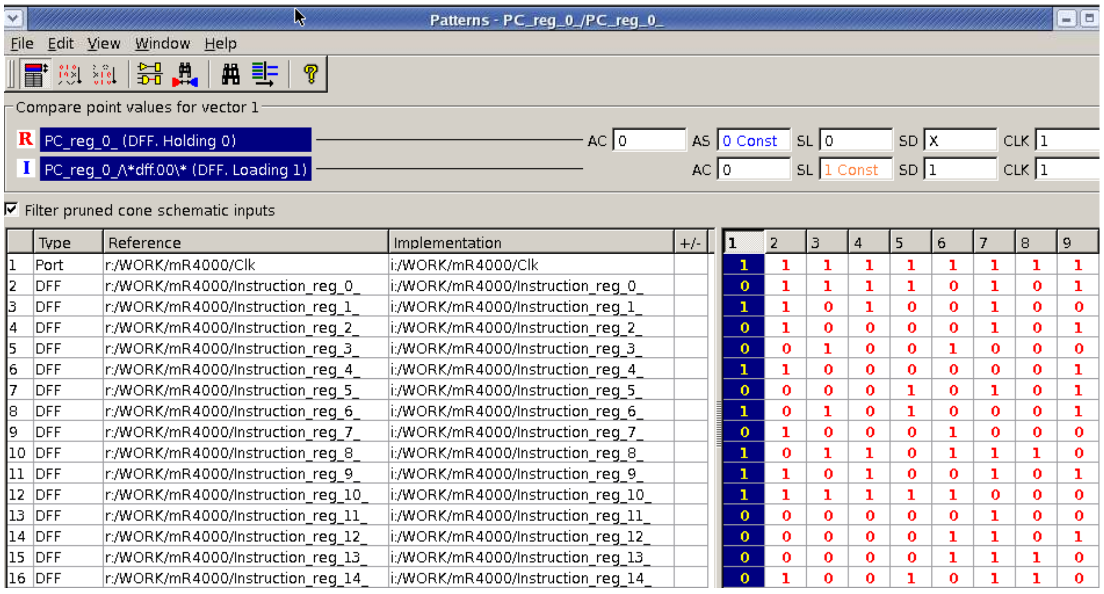
```
 Exists in the ref cone but not in the impl cone for 1 compare point(s):
      i:/WORK/mR4000/cntrl_state_reg_7_

----------
-------------------------------
*****************************************************************************************
Analysis Completed
1
```

Formality (verify)>

Ready   Shell State: verify

2d) Pattern Viewer:

View the pattern window of one of the failing compare points. Notice that the logic just seems to be different even though the logic cone inputs are the same:



Compare point values for vector 1

R	PC_reg_0_ (DFF. Holding 0)	AC	0	AS	0 Const	SL	0	SD	X	CLK	1
I	PC_reg_0_/*dff.00* (DFF. Loading 1)	AC	0			SL	1 Const	SD	1	CLK	1

☑ Filter pruned cone schematic inputs

	Type	Reference	Implementation	+/-	1	2	3	4	5	6	7	8	9
1	Port	r:/WORK/mR4000/Clk	i:/WORK/mR4000/Clk		1	1	1	1	1	1	1	1	1
2	DFF	r:/WORK/mR4000/Instruction_reg_0_	i:/WORK/mR4000/Instruction_reg_0_		0	1	1	1	1	0	1	0	1
3	DFF	r:/WORK/mR4000/Instruction_reg_1_	i:/WORK/mR4000/Instruction_reg_1_		1	1	0	1	0	0	1	0	0
4	DFF	r:/WORK/mR4000/Instruction_reg_2_	i:/WORK/mR4000/Instruction_reg_2_		0	1	0	0	0	0	1	0	1
5	DFF	r:/WORK/mR4000/Instruction_reg_3_	i:/WORK/mR4000/Instruction_reg_3_		0	0	1	0	0	1	0	0	0
6	DFF	r:/WORK/mR4000/Instruction_reg_4_	i:/WORK/mR4000/Instruction_reg_4_		1	1	0	0	0	0	0	0	1
7	DFF	r:/WORK/mR4000/Instruction_reg_5_	i:/WORK/mR4000/Instruction_reg_5_		0	0	0	0	1	0	1	0	1
8	DFF	r:/WORK/mR4000/Instruction_reg_6_	i:/WORK/mR4000/Instruction_reg_6_		1	0	1	0	1	0	0	0	1
9	DFF	r:/WORK/mR4000/Instruction_reg_7_	i:/WORK/mR4000/Instruction_reg_7_		0	1	0	0	0	1	0	0	0
10	DFF	r:/WORK/mR4000/Instruction_reg_8_	i:/WORK/mR4000/Instruction_reg_8_		1	0	1	1	0	1	1	1	0
11	DFF	r:/WORK/mR4000/Instruction_reg_9_	i:/WORK/mR4000/Instruction_reg_9_		1	1	0	1	0	0	1	0	1
12	DFF	r:/WORK/mR4000/Instruction_reg_10_	i:/WORK/mR4000/Instruction_reg_10_		1	1	1	1	1	1	0	0	0
13	DFF	r:/WORK/mR4000/Instruction_reg_11_	i:/WORK/mR4000/Instruction_reg_11_		0	0	0	0	0	0	1	0	0
14	DFF	r:/WORK/mR4000/Instruction_reg_12_	i:/WORK/mR4000/Instruction_reg_12_		0	0	0	0	0	1	1	0	1
15	DFF	r:/WORK/mR4000/Instruction_reg_13_	i:/WORK/mR4000/Instruction_reg_13_		0	0	0	0	0	1	1	1	0
16	DFF	r:/WORK/mR4000/Instruction_reg_14_	i:/WORK/mR4000/Instruction_reg_14_		0	1	0	0	1	0	1	1	0

In many situations where one compare point is holding a value while the other compare point is loading a different value, this is an indication of RTL interpretation differences between Formality and Design Compiler. It is likely that something like parallel/full case pragma interpretation is making a difference in the verification of this design.

3.) Resolve the RTL interpretation issue. You can set these variables:

```
set synopsys_auto_setup true
```

Or, set these:

```
set hdlin_ignore_parallel_case  false
set hdlin_ignore_full_case  false
set hdlin_error_on_mismatch_message false
```

4.) Fix up the FM TCL script and re-run verification. If you do not get a successful verification, view the .solution directory.

FM Lab3: Scan Mode

Objective: This lab contains an implementation design with scan and clock-gating inserted. You must change the "runme.fms" FM TCL script to get a successful verification.

Lab flow:

1.) Run the verification using the existing "runme.fms" script.

2.) Find clues to indicate potential problems and fix them.

2a) Transcript messages:

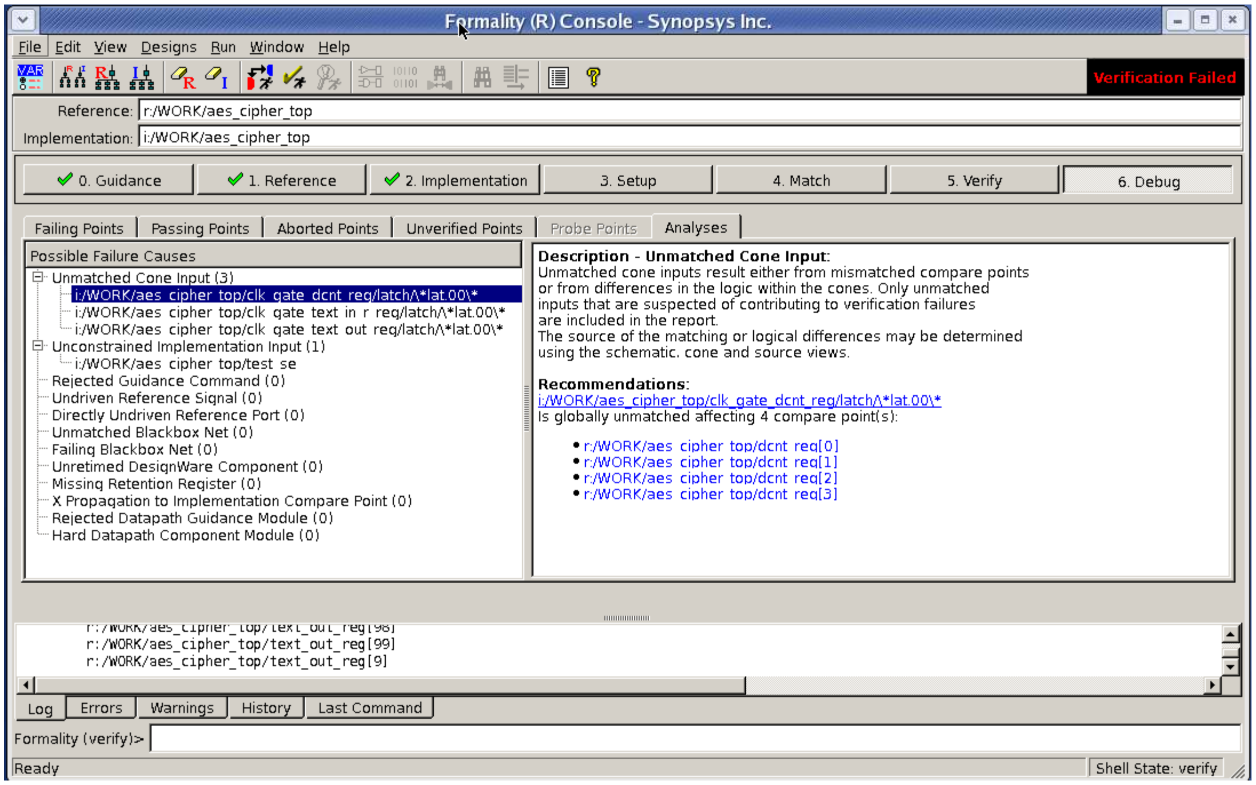
Formality debugging involves collecting information that may point to the reason why the design fails verification. Always review the transcript messages first.

Note the SVF Guidance Summary in the transcript:

```
***************************** Guidance Summary ****************************
                                          Status
Command                 Accepted   Rejected  Unsupported  Unprocessed  Total
----------------------------------------------------------------------------
change_names        :          2          0            0            0      2
environment         :          4          0            0            0      4
instance_map        :          3          0            0            0      3
mark                :          4          0            0            0      4
reg_constant        :         24          0            0            0     24
scan_input          :          0          1            0            0      1
uniquify            :         33          0            0            0     33
```

Unless the Auto Setup Mode is enabled, Formality will ignore scan_input guidance found in the SVF file. You will need to disable scan mode in your FM Tcl script. Continue on with the debugging first.

2b) Using the GUI, run “Analyze” on the failing compare points. You will see the following:



Description - Unmatched Cone Input:
Unmatched cone inputs result either from mismatched compare points or from differences in the logic within the cones. Only unmatched inputs that are suspected of contributing to verification failures are included in the report.
The source of the matching or logical differences may be determined using the schematic, cone and source views.

Recommendations:
i:/WORK/aes_cipher_top/clk_gate_dcnt_reg/latch/*lat.00*
Is globally unmatched affecting 4 compare point(s):

- r:/WORK/aes_cipher_top/dcnt_reg[0]
- r:/WORK/aes_cipher_top/dcnt_reg[1]
- r:/WORK/aes_cipher_top/dcnt_reg[2]
- r:/WORK/aes_cipher_top/dcnt_reg[3]

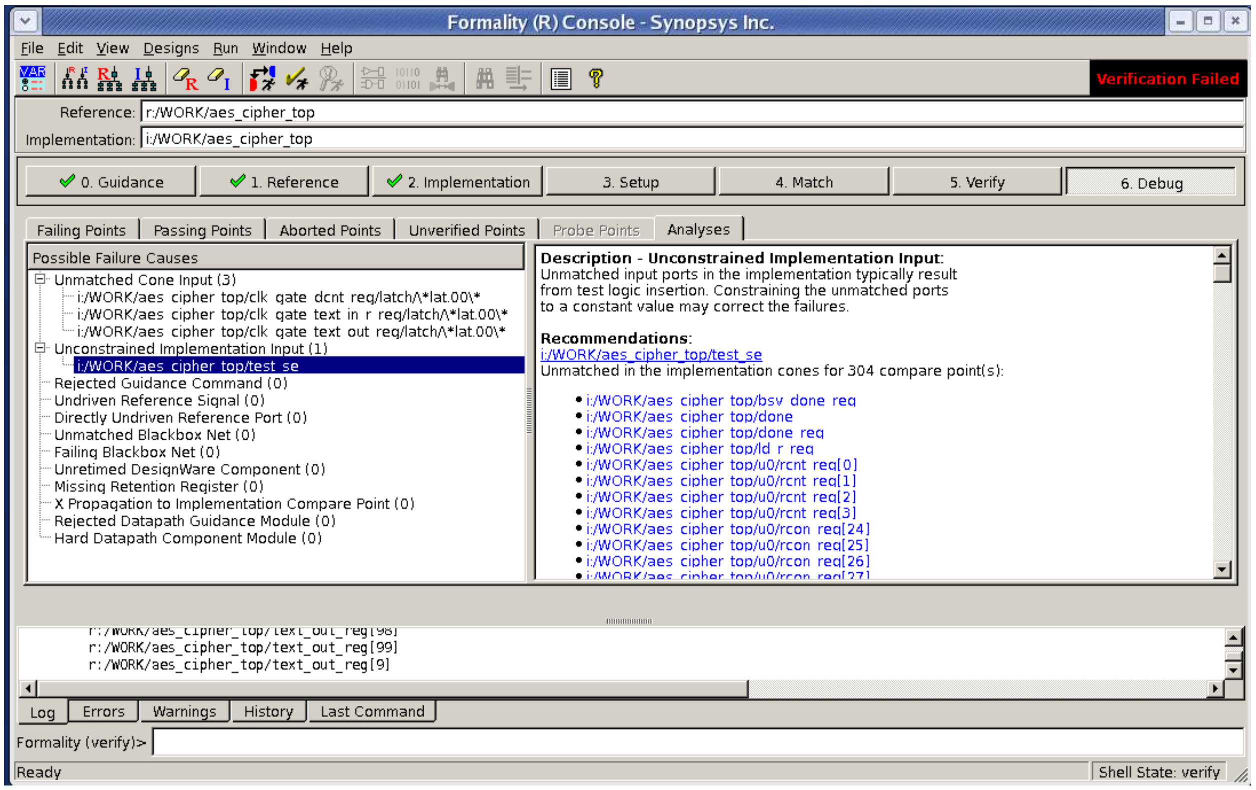
```
Possible Failure Causes
Unmatched Cone Input (3)
    i:/WORK/aes_cipher_top/clk_gate_dcnt_reg/latch/\*lat.00\*
    i:/WORK/aes_cipher_top/clk_gate_text_in_r_reg/latch/\*lat.00\*
    i:/WORK/aes_cipher_top/clk_gate_text_out_reg/latch/\*lat.00\*
Unconstrained Implementation Input (1)
    i:/WORK/aes_cipher_top/test_se
Rejected Guidance Command (0)
Undriven Reference Signal (0)
Directly Undriven Reference Port (0)
Unmatched Blackbox Net (0)
Failing Blackbox Net (0)
Unretimed DesignWare Component (0)
Missing Retention Register (0)
X Propagation to Implementation Compare Point (0)
Rejected Datapath Guidance Module (0)
Hard Datapath Component Module (0)
```

```
r:/WORK/aes_cipher_top/text_out_reg[98]
r:/WORK/aes_cipher_top/text_out_reg[99]
r:/WORK/aes_cipher_top/text_out_reg[9]
```

Reference: r:/WORK/aes_cipher_top
Implementation: i:/WORK/aes_cipher_top
Formality (verify)>
Shell State: verify

There appear to be two different problems with this verification.

The first is a list of unmatched implementation latches that are named "clk_gate_....". This is an indication that clock-gating is in the design. Since these are not recognized as clock-gating latches, it is probable that the variable "verification_clock_gate_hold_mode" was not set. So, Formality is treating these latches as compare points, and failing the verification.



Reference: r:/WORK/aes_cipher_top
Implementation: i:/WORK/aes_cipher_top

```
Possible Failure Causes
Unmatched Cone Input (3)
   i:/WORK/aes_cipher_top/clk_gate_dcnt_reg/latch/\*lat.00\*
   i:/WORK/aes_cipher_top/clk_gate_text_in_r_reg/latch/\*lat.00\*
   i:/WORK/aes_cipher_top/clk_gate_text_out_reg/latch/\*lat.00\*
Unconstrained Implementation Input (1)
   i:/WORK/aes_cipher_top/test_se
Rejected Guidance Command (0)
Undriven Reference Signal (0)
Directly Undriven Reference Port (0)
Unmatched Blackbox Net (0)
Failing Blackbox Net (0)
Unretimed DesignWare Component (0)
Missing Retention Register (0)
X Propagation to Implementation Compare Point (0)
Rejected Datapath Guidance Module (0)
Hard Datapath Component Module (0)
```

Description - Unconstrained Implementation Input:
Unmatched input ports in the implementation typically result from test logic insertion. Constraining the unmatched ports to a constant value may correct the failures.

Recommendations:
i:/WORK/aes_cipher_top/test_se
Unmatched in the implementation cones for 304 compare point(s):

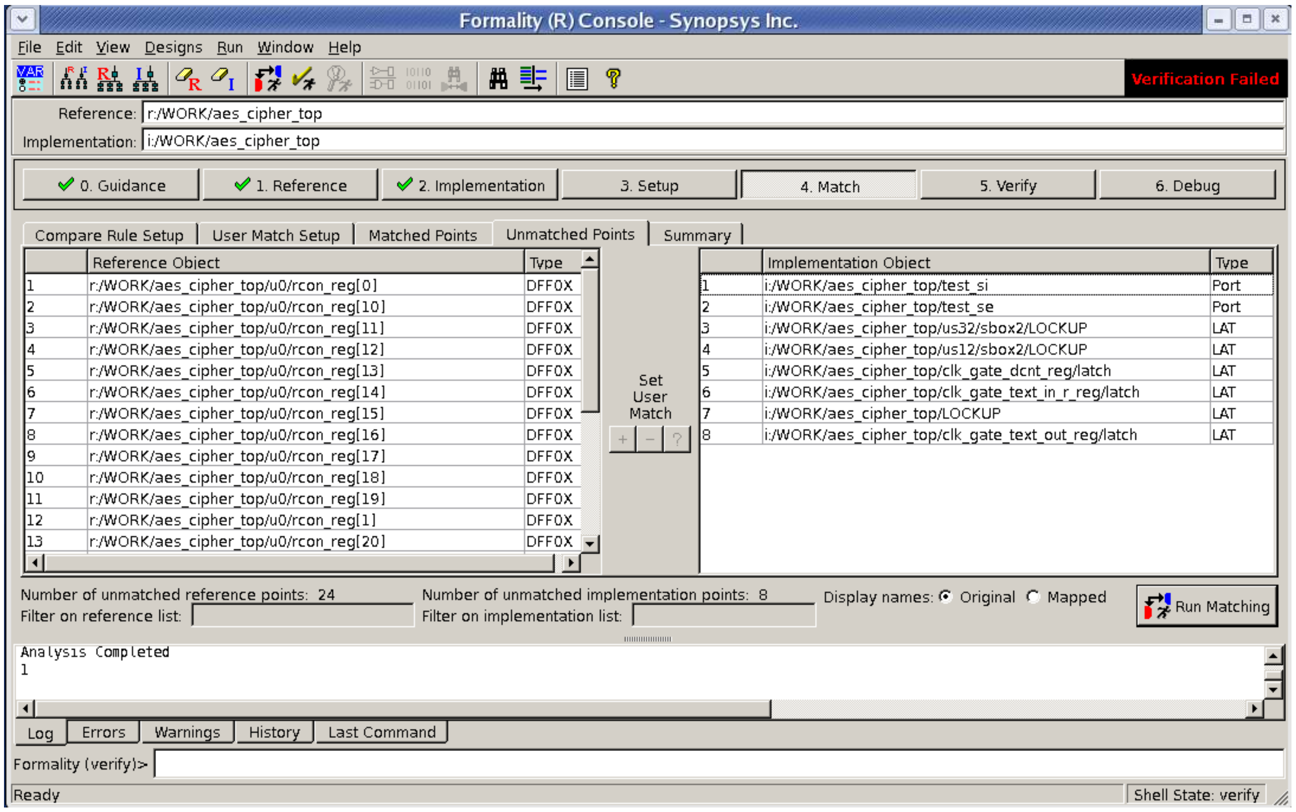
- i:/WORK/aes_cipher_top/bsv_done_reg
- i:/WORK/aes_cipher_top/done
- i:/WORK/aes_cipher_top/done_reg
- i:/WORK/aes_cipher_top/ld_r_reg
- i:/WORK/aes_cipher_top/u0/rcnt_reg[0]
- i:/WORK/aes_cipher_top/u0/rcnt_reg[1]
- i:/WORK/aes_cipher_top/u0/rcnt_reg[2]
- i:/WORK/aes_cipher_top/u0/rcnt_reg[3]
- i:/WORK/aes_cipher_top/u0/rcon_reg[24]
- i:/WORK/aes_cipher_top/u0/rcon_reg[25]
- i:/WORK/aes_cipher_top/u0/rcon_reg[26]
- i:/WORK/aes_cipher_top/u0/rcon_reg[27]

```
r:/WORK/aes_cipher_top/text_out_reg[98]
r:/WORK/aes_cipher_top/text_out_reg[99]
r:/WORK/aes_cipher_top/text_out_reg[9]
```

Formality (verify)>

The second issue is an unconstrained implementation input “test_se”. From this we can assume that no constant was set on the test input to disable scan mode. This needs to be done to have a successful verification.

3.) Before changing the FM Tcl script, view the Match tab. Look at the unmatched points listed for the implementation design.



Reference: r:/WORK/aes_cipher_top
Implementation: i:/WORK/aes_cipher_top

	Reference Object	Type
1	r:/WORK/aes_cipher_top/u0/rcon_reg[0]	DFF0X
2	r:/WORK/aes_cipher_top/u0/rcon_reg[10]	DFF0X
3	r:/WORK/aes_cipher_top/u0/rcon_reg[11]	DFF0X
4	r:/WORK/aes_cipher_top/u0/rcon_reg[12]	DFF0X
5	r:/WORK/aes_cipher_top/u0/rcon_reg[13]	DFF0X
6	r:/WORK/aes_cipher_top/u0/rcon_reg[14]	DFF0X
7	r:/WORK/aes_cipher_top/u0/rcon_reg[15]	DFF0X
8	r:/WORK/aes_cipher_top/u0/rcon_reg[16]	DFF0X
9	r:/WORK/aes_cipher_top/u0/rcon_reg[17]	DFF0X
10	r:/WORK/aes_cipher_top/u0/rcon_reg[18]	DFF0X
11	r:/WORK/aes_cipher_top/u0/rcon_reg[19]	DFF0X
12	r:/WORK/aes_cipher_top/u0/rcon_reg[1]	DFF0X
13	r:/WORK/aes_cipher_top/u0/rcon_reg[20]	DFF0X

	Implementation Object	Type
1	i:/WORK/aes_cipher_top/test_si	Port
2	i:/WORK/aes_cipher_top/test_se	Port
3	i:/WORK/aes_cipher_top/us32/sbox2/LOCKUP	LAT
4	i:/WORK/aes_cipher_top/us12/sbox2/LOCKUP	LAT
5	i:/WORK/aes_cipher_top/clk_gate_dcnt_reg/latch	LAT
6	i:/WORK/aes_cipher_top/clk_gate_text_in_r_reg/latch	LAT
7	i:/WORK/aes_cipher_top/LOCKUP	LAT
8	i:/WORK/aes_cipher_top/clk_gate_text_out_reg/latch	LAT

Number of unmatched reference points: 24
Filter on reference list:
Number of unmatched implementation points: 8
Filter on implementation list:
Display names: [✓ Original] Mapped

```
Analysis Completed
1
```

Formality (verify)>
Ready
Shell State: verify

Again, note the unmatched input port “test_se” in the implementation design. This is an indication of scan in the netlist.

Also note that the “clk_gate_...” latches are of type “LAT” and not “CGLAT” denoting clock-gating latches.

4.) Set a constant on the test input as recommended by the analyze points command. Also, set the clock-gating variable to “low”.

```
setup
set_constant i:/WORK/aes_cipher_top/test_se 0
set verification_clock_gate_hold_mode low
verify
```

5.) Fix up the FM TCL script and re-run verification. If you do not get a successful verification, view the .solution directory.

6.) Note that if you use the Auto Setup Mode with “set synopsys_auto_setup true”, Formality will automatically disable scan and turn on clock-gating. You would not have to do anything else for setup.

FM Lab4: Constant Register Recognition (Modifying SVF File)

Objective: This lab requires you to modify the SVF file so that Formality can recognize a constant register to successfully verify the design.

Note that this is a design contrived specifically to show an example of a naming issue between DC and Formality.

Key ideas:

a.) Practice using these Formality SVF debugging command:

- "analyze_points" with the option "-failing"
- "report_svf_operation" with the options
  "-summary"
  "-status rejected"
  "compare_point(s)"

b.) Practice modifying an ASCII SVF file:

- Use your favorite text editor to replace DC name in SVF with the corresponding Formality name of the potentially constant register

Lab flow:

1.) Run the verification using the existing "runme.fms" script.

2.) Find clues to indicate potential problems.

2a) Transcript messages:

Formality debugging involves collecting information that may point to the reason why the design fails verification. Always review the transcript messages first.

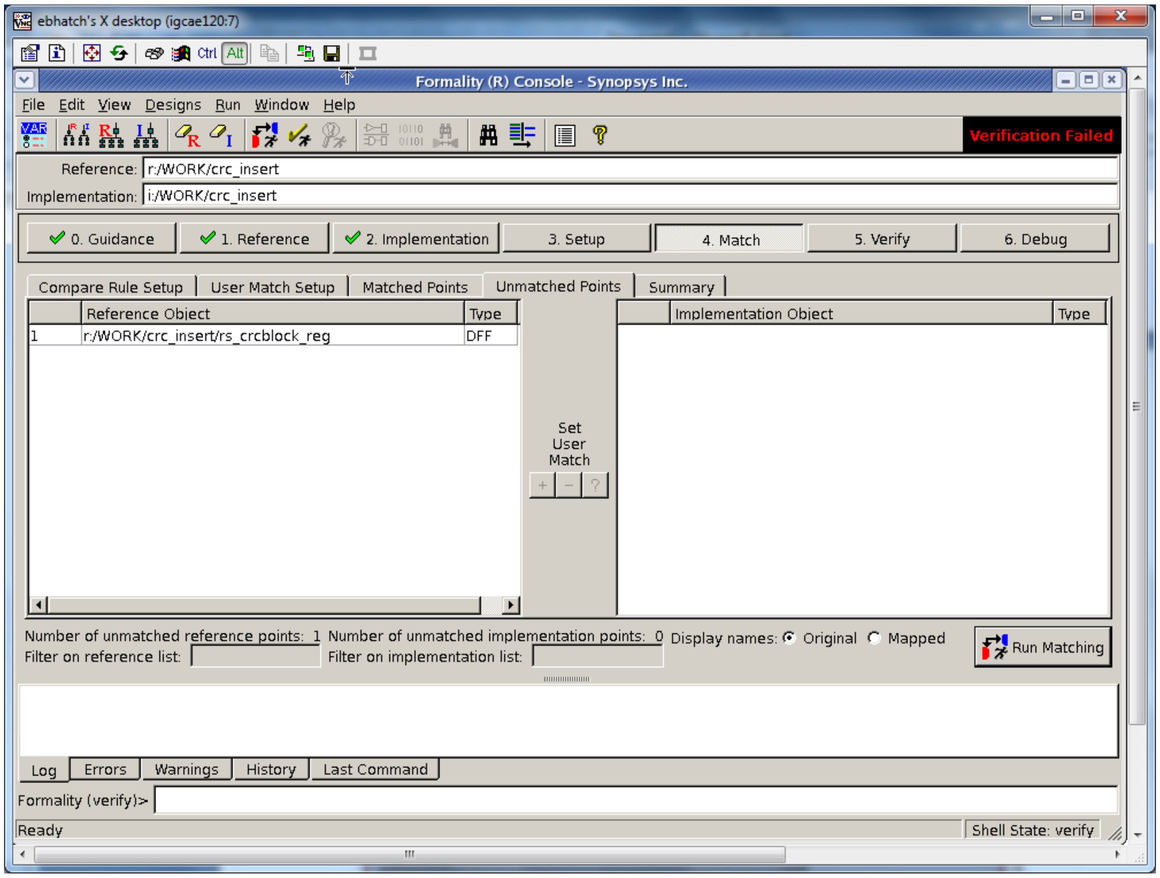
Note the SVF Guidance Summary in the transcript:

```
***************************** Guidance Summary *****************************
                                          Status
Command                   Accepted   Rejected  Unsupported  Unprocessed  Total
----------------------------------------------------------------------------
change_names          :          7          0          0          0          7
environment           :          3          0          0          0          3
instance_map          :          2          0          0          0          2
mark                  :          2          0          0          0          2
reg_constant          :          0          1          0          0          1
uniquify              :          2          0          0          0          2
```

There is 1 rejected SVF guide_reg_constant operation. For several designs, this may be fine. Formality can figure out most constant registers; however, for this lab, this testcase will fail verification because this register is not recognized as a constant register.

This Guidance Summary table can be produced anytime after matching by using the command “report_guidance -summary”.

2b) Here is a picture of the GUI unmatched points tab. Notice the single unmatched register in the reference design that does not exist in the implementation design. Sometimes this is fine. We need to confirm with using either the “analyze” command or the pattern viewer to see if this register contributes to failing compare points.



Reference: r:/WORK/crc_insert
Implementation: i:/WORK/crc_insert

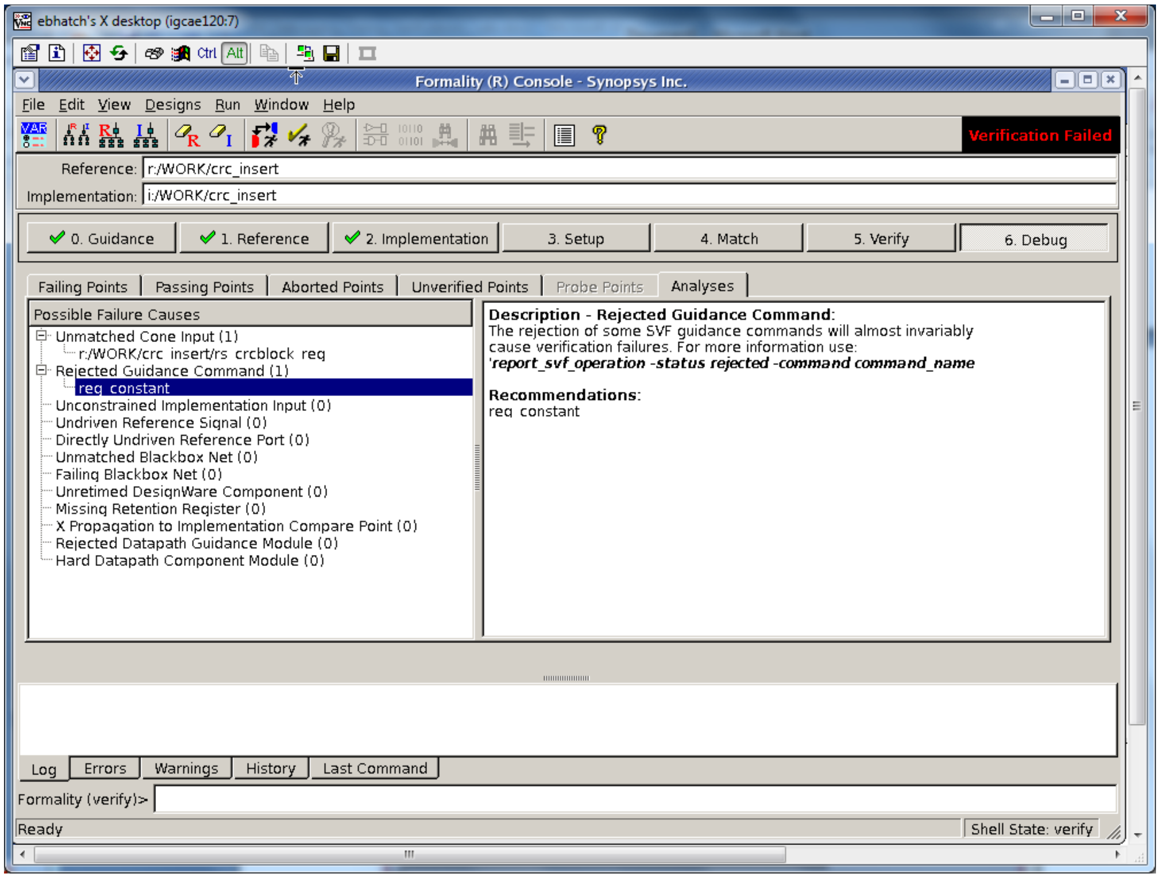
	Reference Object	Type
1	r:/WORK/crc_insert/rs_crcblock_reg	DFF

	Implementation Object	Type

Number of unmatched reference points: 1 Number of unmatched implementation points: 0 Display names: [✓ Original] Mapped
Filter on reference list: Filter on implementation list:

Formality (verify)>
Ready
Shell State: verify

2c) Run “analyze -failing” on the testcase.



Reference: r:/WORK/crc_insert
Implementation: i:/WORK/crc_insert

Possible Failure Causes
- Unmatched Cone Input (1)
  - r:/WORK/crc insert/rs crcblock reg
- Rejected Guidance Command (1)
  - reg constant
- Unconstrained Implementation Input (0)
- Undriven Reference Signal (0)
- Directly Undriven Reference Port (0)
- Unmatched Blackbox Net (0)
- Failing Blackbox Net (0)
- Unretimed DesignWare Component (0)
- Missing Retention Register (0)
- X Propagation to Implementation Compare Point (0)
- Rejected Datapath Guidance Module (0)
- Hard Datapath Component Module (0)

Description - Rejected Guidance Command:
The rejection of some SVF guidance commands will almost invariably cause verification failures. For more information use:
'report_svf_operation -status rejected -command command_name

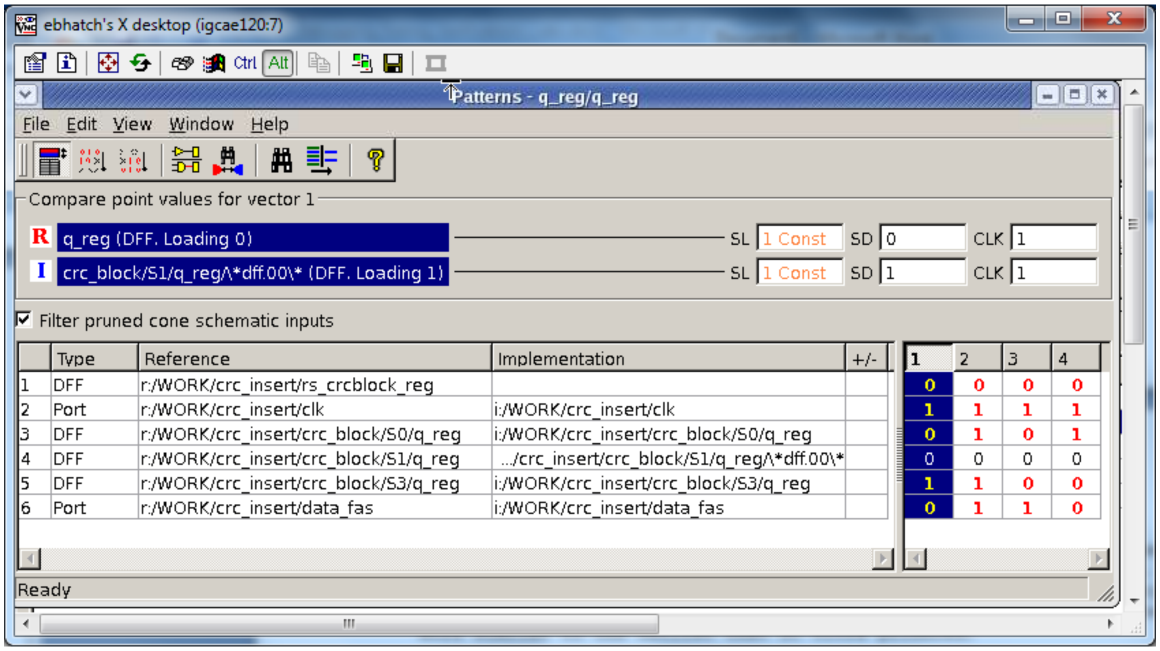
Recommendations:
reg constant

Formality (verify)>

```
fm_shell (verify)> analyze_points -failing

*********************************** Analysis Results
***********************************
Found 1 Unmatched Cone Input
--------------------------------
Unmatched cone inputs result either from mismatched compare points
or from differences in the logic within the cones. Only unmatched
inputs that are suspected of contributing to verification failures
are included in the report.
The source of the matching or logical differences may be determined
using the schematic, cone and source views.
--------------------------------
r:/WORK/crc_insert/rs_crcblock_reg
    Is globally unmatched affecting 4 compare point(s):
        i:/WORK/crc_insert/crc_block/S1/q_reg
        i:/WORK/crc_insert/crc_block/S2/q_reg
        i:/WORK/crc_insert/crc_block/S3/q_reg
        i:/WORK/crc_insert/reset_crc4

-----------
--------------------------------
Found 1 Rejected Guidance Command
--------------------------------
The rejection of some SVF guidance commands will almost invariably
cause verification failures. For more information use:
        'report_svf_operation -status rejected -command command_name
--------------------------------
reg_constant
-----------
--------------------------------
*********************************************************************************
******
```

Notice the suggestion to run the command report_svf_operation. We will do that after just a few more steps.

3.) Let’s take a quick view of the failing patterns:



Compare point values for vector 1

R q_reg (DFF. Loading 0) — SL 1 Const SD 0 CLK 1

I crc_block/S1/q_reg/*dff.00* (DFF. Loading 1) — SL 1 Const SD 1 CLK 1

☑ Filter pruned cone schematic inputs

	Type	Reference	Implementation	+/-	1	2	3	4
1	DFF	r:/WORK/crc_insert/rs_crcblock_reg			0	0	0	0
2	Port	r:/WORK/crc_insert/clk	i:/WORK/crc_insert/clk		1	1	1	1
3	DFF	r:/WORK/crc_insert/crc_block/S0/q_reg	i:/WORK/crc_insert/crc_block/S0/q_reg		0	1	0	1
4	DFF	r:/WORK/crc_insert/crc_block/S1/q_reg	.../crc_insert/crc_block/S1/q_reg/*dff.00*		0	0	0	0
5	DFF	r:/WORK/crc_insert/crc_block/S3/q_reg	i:/WORK/crc_insert/crc_block/S3/q_reg		1	1	0	0
6	Port	r:/WORK/crc_insert/data_fas	i:/WORK/crc_insert/data_fas		0	1	1	0

The patterns indicate that single unmatched reference register has a value of “0” for every failing pattern. Perhaps it is a constant1 register?

4.) Now, let’s look at the reason for Formality rejecting the SVF reg_constant guidance.

```
fm_shell (verify)> report_svf_operation -status rejected -command reg_constant

## SVF Operation 11 (Line: 82) - reg_constant.  Status: rejected
## Operation Id: 11
guide_reg_constant \
  -design { crc_insert } \
  { rs_crcblk_reg } 1

Info:  guide_reg_constant 11 (Line: 82) Cannot find master reference cell
'rs_crcblk_reg'.
```

Formality cannot find the register named “rs_crcblk_reg” in the reference design it created. However, the unmatched register in the reference design is named “rs_crcblock_reg”. The names are close, but do not completely match up, so Formality rejected the guidance. The problem could be a naming concordance difference between Design Compiler and Formality when each tool created the design from the RTL.

5.) Modify the svf.txt file located under "formality_svf" directory that was automatically generated during the "set_svf crc_insert.svf" command. Change the name of the register in the SVF from "rs_crcblk_reg" to "rs_crcblock_reg" which Formality will recognize in its reference container.

6.) After modifying the svf.txt file, rename the directory from "formality_svf" to "modified_svf". Change the Formality TCL script to point to the new directory. Formality will look for SVF files in the specified directory.

7.) After running with the modified SVF file, the Guidance Summary should be clean. You should have a successful verification.

```
***************************** Guidance Summary ****************************
                                         Status
Command                 Accepted   Rejected  Unsupported  Unprocessed  Total
----------------------------------------------------------------------------
change_names        :          7          0          0          0          7
environment         :          3          0          0          0          3
instance_map        :          2          0          0          0          2
mark                :          2          0          0          0          2
reg_constant        :          1          0          0          0          1
uniquify            :          2          0          0          0          2
```

Formality will perform an internal verification check on the potentially constant register before accepting reg_constant guidance.

8.) Instead of modifying the SVF, you could have verified this register against a constant1, and found that it was truly a constant 1. Then, you could use the set_constant command to manually set this register to a constant1 value. This will also give a successful verification result.

FM Lab5: Recognizing Clock-gating

Objective: This is a gate_vs_gate verification. This testcase has clock-gating circuitry in both the reference and implementation designs. You must change the "runme.fms" FM TCL script to get a successful verification.

Note: The legacy variable *verification_clock_gate_hold_mode* turns on functionality that identifies clock-gating circuitry leading to a clk-pin of a rising edge DFF.

Lab flow:

1.) Run the verification using the existing "runme.fms" script.

2.) Find clues to indicate potential problems.

2a) Transcript messages:

The only message is the name of the failing compare point:

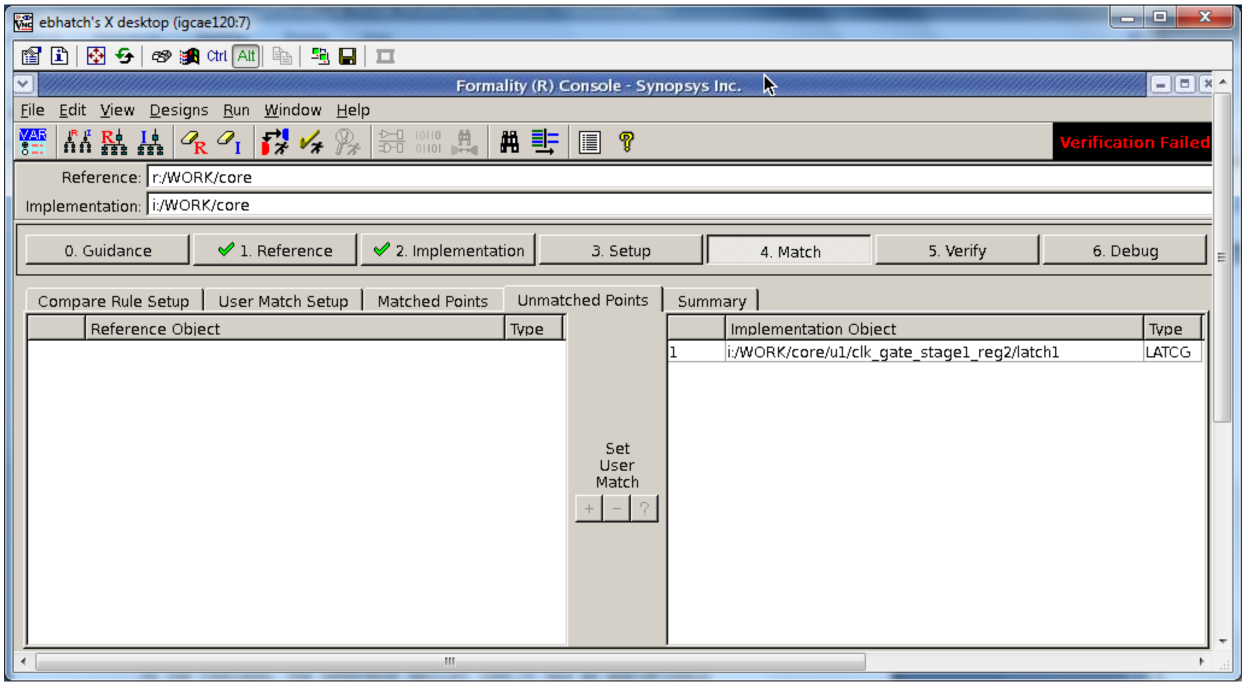
```
Status:  Verifying...
    Compare point u1/LOCKUP failed (is not equivalent)
```

Lockup latches are usually positioned at the end of the clock-gating chain during clock tree synthesis.

Let’s investigate this failure further.

2b) Messages from matching:

Bring up the GUI and view the Match tab. Here is a picture of the GUI unmatched points tab.



Reference: r:/WORK/core
Implementation: i:/WORK/core

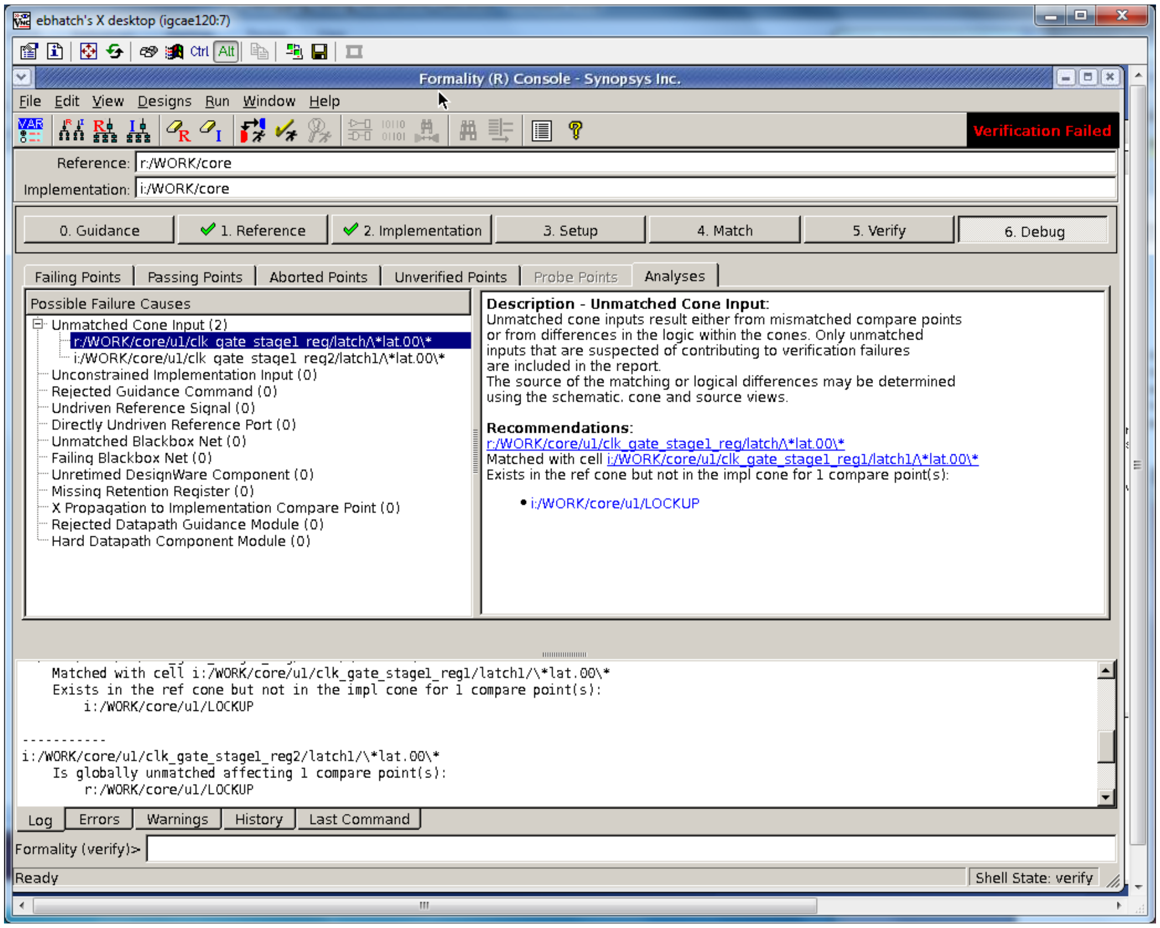
	Reference Object	Type

	Implementation Object	Type
1	i:/WORK/core/u1/clk_gate_stage1_reg2/latch1	LATCG

Remember that this is a gate_vs_gate design with clk-gating latches in both the reference and implementation. Here we see an extra clock-gating latch in the implementation only.

It is common for a clock-gating cell to be split into multiple clock-gating cells to satisfy fanout requirements and skew requirements during clock-tree synthesis.

2c) Run "analyze –failing" on the testcase.



Reference: r:/WORK/core
Implementation: i:/WORK/core

Possible Failure Causes
- Unmatched Cone Input (2)
  - r:/WORK/core/u1/clk_gate_stage1_reg/latch/*lat.00*
  - i:/WORK/core/u1/clk_gate_stage1_reg2/latch1/*lat.00*
- Unconstrained Implementation Input (0)
- Rejected Guidance Command (0)
- Undriven Reference Signal (0)
- Directly Undriven Reference Port (0)
- Unmatched Blackbox Net (0)
- Failing Blackbox Net (0)
- Unretimed DesignWare Component (0)
- Missing Retention Register (0)
- X Propagation to Implementation Compare Point (0)
- Rejected Datapath Guidance Module (0)
- Hard Datapath Component Module (0)

Description - Unmatched Cone Input:
Unmatched cone inputs result either from mismatched compare points or from differences in the logic within the cones. Only unmatched inputs that are suspected of contributing to verification failures are included in the report.
The source of the matching or logical differences may be determined using the schematic. cone and source views.

Recommendations:
r:/WORK/core/u1/clk_gate_stage1_reg/latch/*lat.00*
Matched with cell i:/WORK/core/u1/clk_gate_stage1_reg1/latch1/*lat.00*
Exists in the ref cone but not in the impl cone for 1 compare point(s):

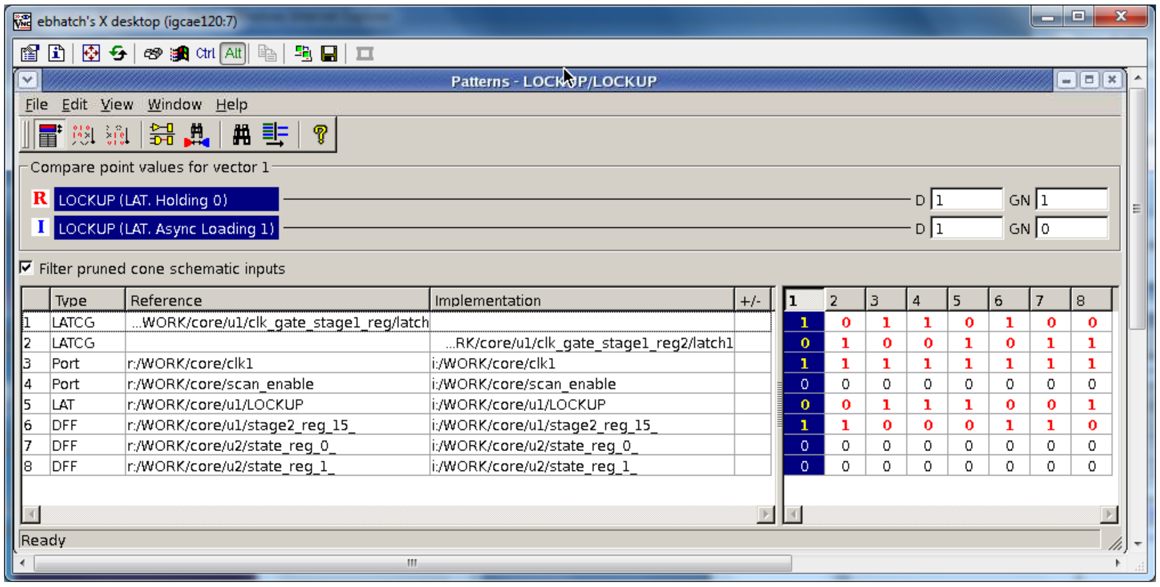
- i:/WORK/core/u1/LOCKUP

```
    Matched with cell i:/WORK/core/u1/clk_gate_stage1_reg1/latch1/\*lat.00\*
    Exists in the ref cone but not in the impl cone for 1 compare point(s):
        i:/WORK/core/u1/LOCKUP

-----------
i:/WORK/core/u1/clk_gate_stage1_reg2/latch1/\*lat.00\*
    Is globally unmatched affecting 1 compare point(s):
        r:/WORK/core/u1/LOCKUP
```

Formality (verify)>

Here we see some hints about what is happening. The failing compare point r:/WORK/core/u1/LOCKUP is affected by a globally unmatched latch i:/WORK/core/u1/clk_gate_stage1_reg2/latch1.

3.) Viewing the pattern viewer for the failing compare point:



Compare point values for vector 1

R	LOCKUP (LAT. Holding 0)	D	1	GN	1
I	LOCKUP (LAT. Async Loading 1)	D	1	GN	0

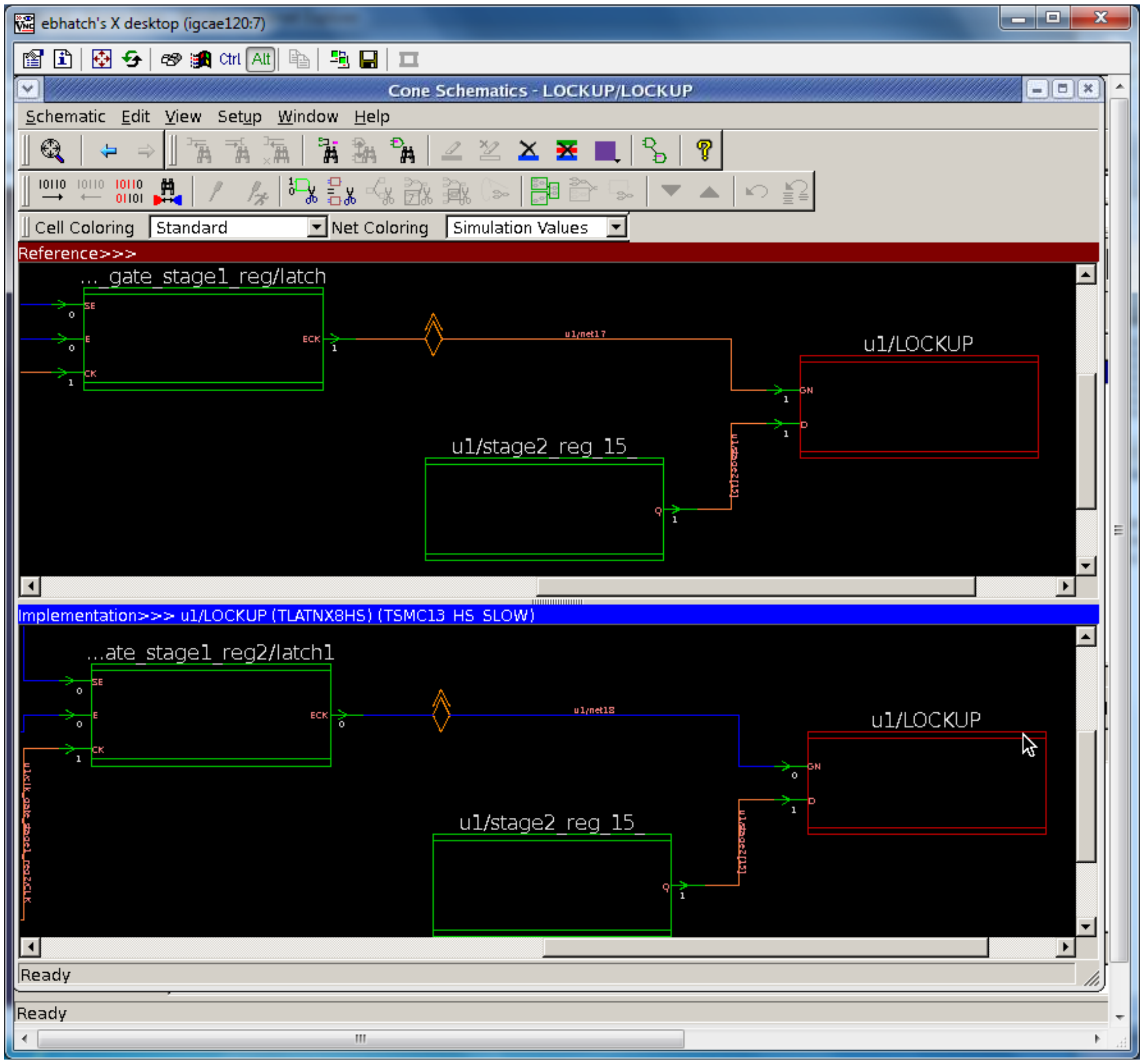
☑ Filter pruned cone schematic inputs

	Type	Reference	Implementation	+/-	1	2	3	4	5	6	7	8
1	LATCG	...WORK/core/u1/clk_gate_stage1_reg/latch			1	0	1	1	0	1	0	0
2	LATCG		...RK/core/u1/clk_gate_stage1_reg2/latch1		0	1	0	0	1	0	1	1
3	Port	r:/WORK/core/clk1	i:/WORK/core/clk1		1	1	1	1	1	1	1	1
4	Port	r:/WORK/core/scan_enable	i:/WORK/core/scan_enable		0	0	0	0	0	0	0	0
5	LAT	r:/WORK/core/u1/LOCKUP	i:/WORK/core/u1/LOCKUP		0	0	1	1	1	0	0	1
6	DFF	r:/WORK/core/u1/stage2_reg_15_	i:/WORK/core/u1/stage2_reg_15_		1	1	0	0	0	1	1	0
7	DFF	r:/WORK/core/u2/state_reg_0_	i:/WORK/core/u2/state_reg_0_		0	0	0	0	0	0	0	0
8	DFF	r:/WORK/core/u2/state_reg_1_	i:/WORK/core/u2/state_reg_1_		0	0	0	0	0	0	0	0

Ready

It appears that even though these latches are clk-gating latches, they are not acting like clock-gating latches for this failing compare point. It appears that the failures are happening if FM places opposite values on them. Normally, clock-gating latches do not act as "active" logic cone inputs.

4.) The logic cones confirm that Formality is placing and using different values on these supposed clock-gating latches:


ebhatch's X desktop (igcae120:7)
Cone Schematics - LOCKUP/LOCKUP
Schematic Edit View Setup Window Help
Cell Coloring Standard Net Coloring Simulation Values
Reference>>>
..._gate_stage1_reg/latch
u1/LOCKUP
u1/stage2_reg_15_
Implementation>>> u1/LOCKUP (TLATNX8HS) (TSMC13_HS_SLOW)
...ate_stage1_reg2/latch1
u1/LOCKUP
u1/stage2_reg_15_
Ready
Ready

At this point you can see that these clk-gating latches are not driving the clk-pin of a rising edge DFF, but rather are going into the lockup latch instead.

The legacy variable verification_clock_gate_hold_mode is not designed to handle this situation.

5.) Our alternative is to try the new clock-gating algorithm invoked by using the variable *verification_clock_gate_edge_analysis*.

6.) Re-run Formality with the new variable to get a successful verification.

FM Lab6: ECO Problem Using Graphical Debugging

Objective: This lab will help you identify and repair a design difference. You must correctly change the gate-level netlist to get a successful verification.

Background: This is an ECO RTL_vs_gate verification. There is no SVF file. A design engineer changed both the RTL and the gate-level netlist manually without re-running synthesis. The modified RTL is correct and golden. It passed simulation with great results.

Your friend, a new engineer, is trying to verify the design and is having problems with debugging the failing verification.

Please help him modify the gate-level netlist to correctly implement the ECO.

Lab flow:

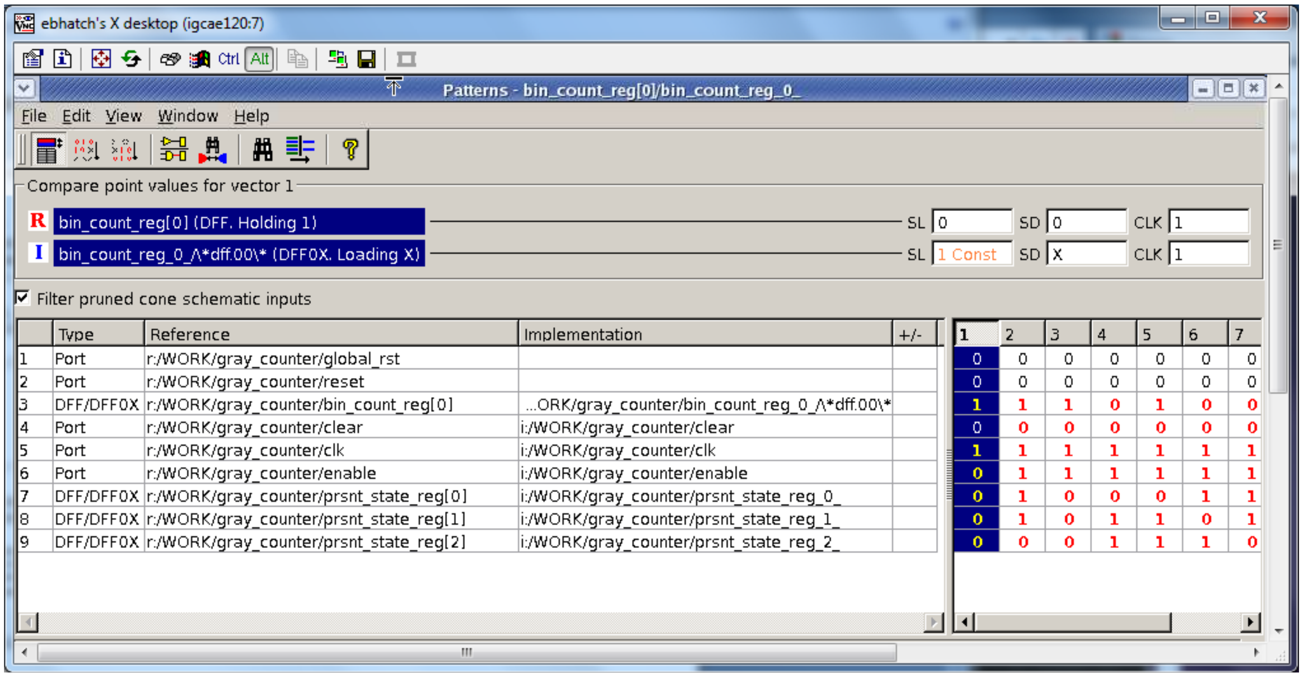
1.) Run the verification using the existing "runme.fms" script.

2.) Find clues to indicate potential problems. The transcript indicates some interesting things in the implementation design:

```
    Warning: 0 (1) undriven nets found in reference (implementation) design; see
formality.log for list (FM-399)
    Info:  0 (2) multiply-driven nets found in reference (implementation) design;
see formality.log for list.
```

3a) The analyze command has no suggestions.

3b) View the pattern window:



Patterns - bin_count_reg[0]/bin_count_reg_0_

Compare point values for vector 1

R bin_count_reg[0] (DFF. Holding 1) SL 0 SD 0 CLK 1

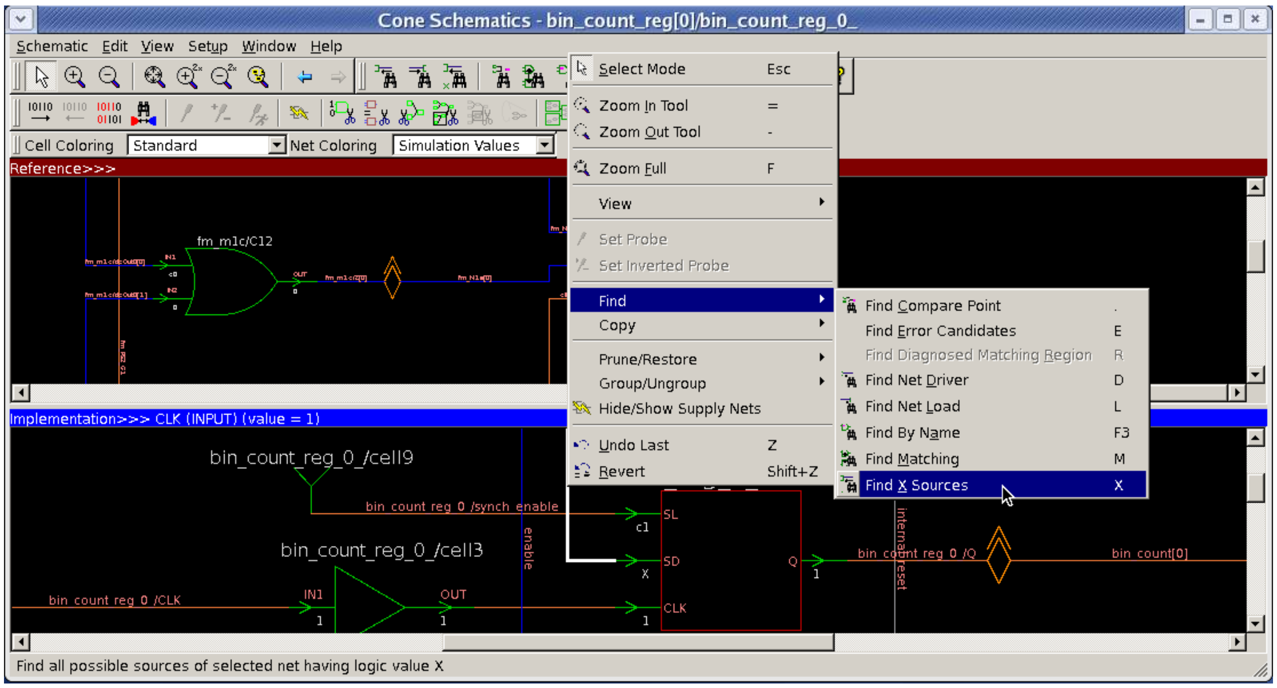
I bin_count_reg_0_/*dff.00* (DFF0X. Loading X) SL 1 Const SD X CLK 1

☑ Filter pruned cone schematic inputs

	Type	Reference	Implementation	+/-	1	2	3	4	5	6	7
1	Port	r:/WORK/gray_counter/global_rst			0	0	0	0	0	0	0
2	Port	r:/WORK/gray_counter/reset			0	0	0	0	0	0	0
3	DFF/DFF0X	r:/WORK/gray_counter/bin_count_reg[0]	...ORK/gray_counter/bin_count_reg_0_/*dff.00*		1	1	1	0	1	0	0
4	Port	r:/WORK/gray_counter/clear	i:/WORK/gray_counter/clear		0	0	0	0	0	0	0
5	Port	r:/WORK/gray_counter/clk	i:/WORK/gray_counter/clk		1	1	1	1	1	1	1
6	Port	r:/WORK/gray_counter/enable	i:/WORK/gray_counter/enable		0	1	1	1	1	1	1
7	DFF/DFF0X	r:/WORK/gray_counter/prsnt_state_reg[0]	i:/WORK/gray_counter/prsnt_state_reg_0_		0	1	0	0	0	1	1
8	DFF/DFF0X	r:/WORK/gray_counter/prsnt_state_reg[1]	i:/WORK/gray_counter/prsnt_state_reg_1_		0	1	0	1	1	0	1
9	DFF/DFF0X	r:/WORK/gray_counter/prsnt_state_reg[2]	i:/WORK/gray_counter/prsnt_state_reg_2_		0	0	0	1	1	1	0

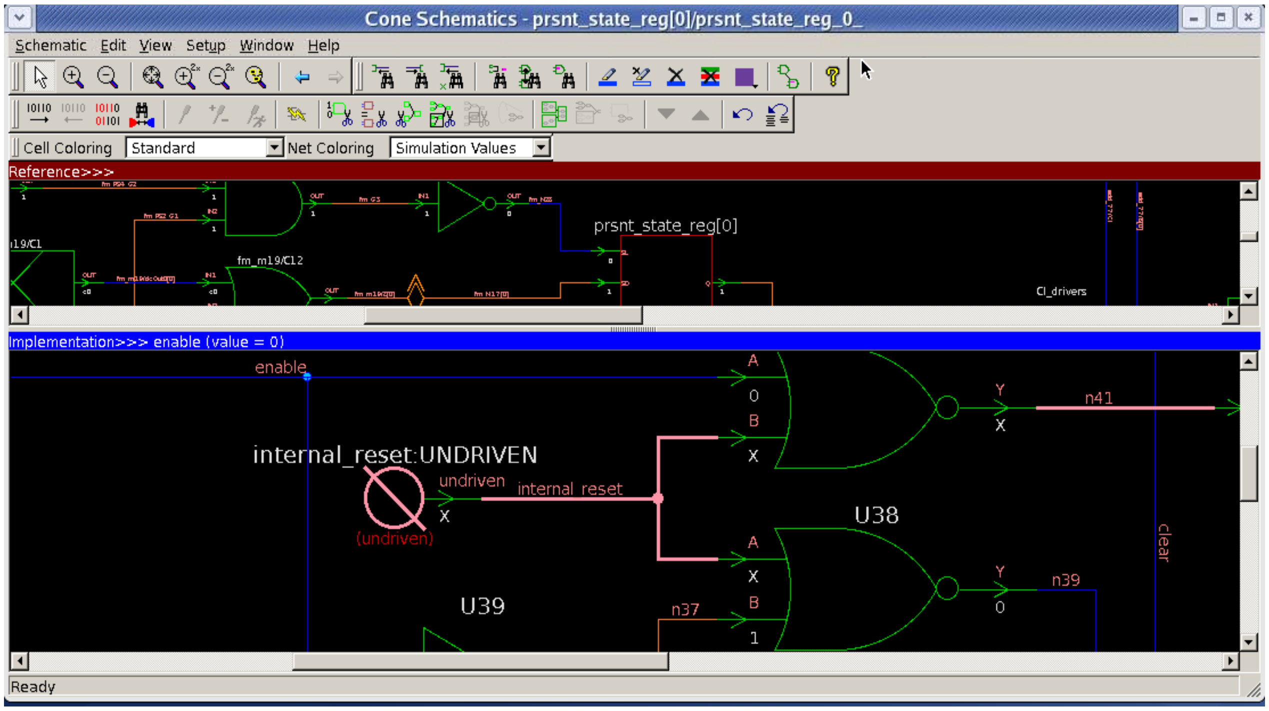
The reference design has two input ports that are named "global_rst" and "reset". These seem to be missing in the logic cone for this compare point. Notice the "X" on the D-input of this implementation register compare point.

4.) View the logic cone to visually inspect the problem.



Find all possible sources of selected net having logic value X

5.) Using the popup menu, perform a "Prune/Restore -> Expand Schematic". Then, select the net with the "X" value in the implementation leading to the SD pin of the DFF. Use the popup menu to "Find X Sources". This will trace back to the source of the "X", highlighting the problem in the netlist.


Cone Schematics - prsnt_state_reg[0]/prsnt_state_reg_0_
Schematic Edit View Setup Window Help
Cell Coloring Standard Net Coloring Simulation Values
Reference>>>
Implementation>>> enable (value = 0)
prsnt_state_reg[0]
enable
internal_reset:UNDRIVEN
undriven
internal_reset
U38
U39
n41
n39
n37
clear
Ready

Note the undriven net in the implementation design. This is the key to the verification difference. The net name is "internal_reset".

6.) Search for this net in the Verilog netlist. You will need to edit the netlist. Use your favorite editor to view and repair the netlist. (FYI - You can bring up a browser window by selecting the component this net drives, bringing up the menu (right mouse click), and selecting View Source. However, cannot edit using this browser.)

You might also want to view the netlist object for this net using the menu item View Object.

7.) Trace this net in the netlist to the components it connects with. Ponder why it is not working and fix it.

8.) Re-run verification. If you do not get a successful verification, view the .solution directory.

FM Lab7: Solving Multiple Problems

Objective: This lab will help you identify and overcome multiple verification problems. You must change the "runme.fms" FM Tcl script to get a successful verification.

Lab flow:

1.) Run the verification using the existing "runme.fms" script.

2.) Find clues to indicate potential problems.

3.) Transcript messages:

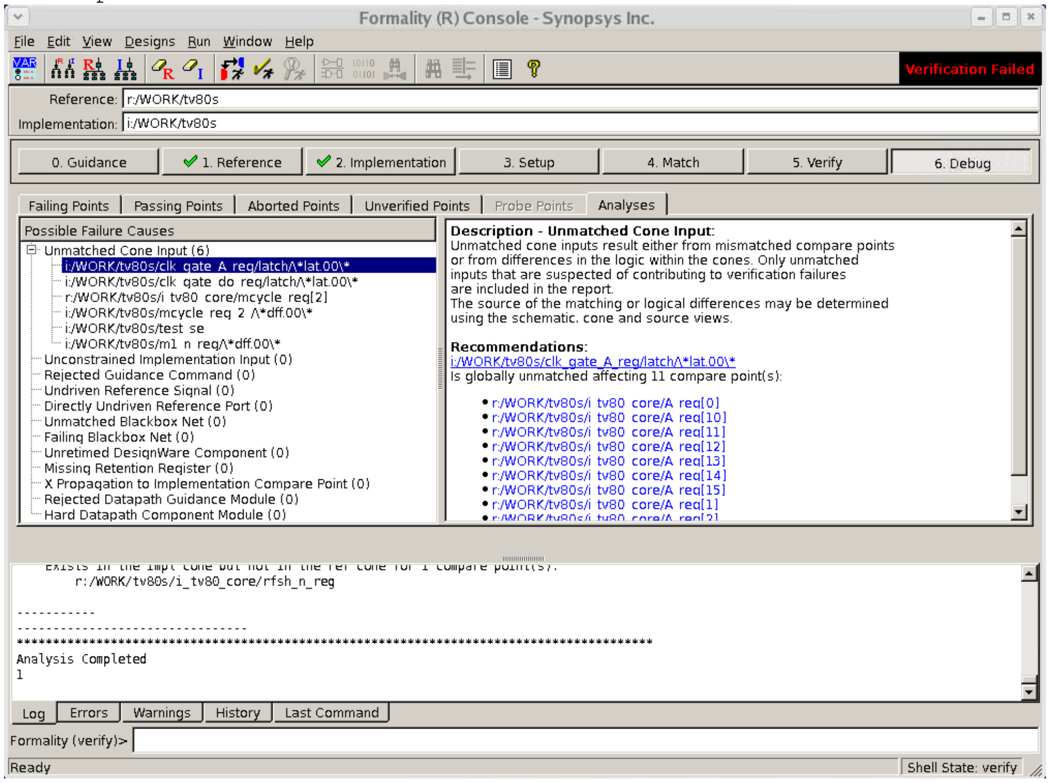
Formality debugging involves collecting information that may point to the reason why the design fails verification. Always look at the transcript messages first.

Note the following messages in the transcript:

```
…
set signature_analysis_match_compare_points false
…
Info:  Formality Guide Files (SVF) can improve matching performance and success by
automating setup.
…
```

If you got this script from another engineer in your company, please ask them why they are turning off signature analysis? Ask them why they are not using the SVF file?

4.) Run "Analyze". Formality indicates that there are logic cone inputs that exist in the implementation design that do not exist in the reference design. This could be caused by clock-gating or by scan, or other problems.



Possible Failure Causes

- Unmatched Cone Input (6)
  - i:/WORK/tv80s/clk gate A req/latch/*lat.00*
  - i:/WORK/tv80s/clk gate do req/latch/*lat.00*
  - r:/WORK/tv80s/i tv80 core/mcycle req[2]
  - i:/WORK/tv80s/mcycle req 2 /*dff.00*
  - i:/WORK/tv80s/test se
  - i:/WORK/tv80s/m1 n req/*dff.00*
- Unconstrained Implementation Input (0)
- Rejected Guidance Command (0)
- Undriven Reference Signal (0)
- Directly Undriven Reference Port (0)
- Unmatched Blackbox Net (0)
- Failing Blackbox Net (0)
- Unretimed DesignWare Component (0)
- Missing Retention Register (0)
- X Propagation to Implementation Compare Point (0)
- Rejected Datapath Guidance Module (0)
- Hard Datapath Component Module (0)

Description - Unmatched Cone Input:
Unmatched cone inputs result either from mismatched compare points or from differences in the logic within the cones. Only unmatched inputs that are suspected of contributing to verification failures are included in the report.
The source of the matching or logical differences may be determined using the schematic. cone and source views.

Recommendations:
i:/WORK/tv80s/clk_gate_A_req/latch/*lat.00*
Is globally unmatched affecting 11 compare point(s):

- r:/WORK/tv80s/i tv80 core/A req[0]
- r:/WORK/tv80s/i tv80 core/A req[10]
- r:/WORK/tv80s/i tv80 core/A req[11]
- r:/WORK/tv80s/i tv80 core/A req[12]
- r:/WORK/tv80s/i tv80 core/A req[13]
- r:/WORK/tv80s/i tv80 core/A req[14]
- r:/WORK/tv80s/i tv80 core/A req[15]
- r:/WORK/tv80s/i tv80 core/A req[1]
- r:/WORK/tv80s/i tv80 core/A req[2]

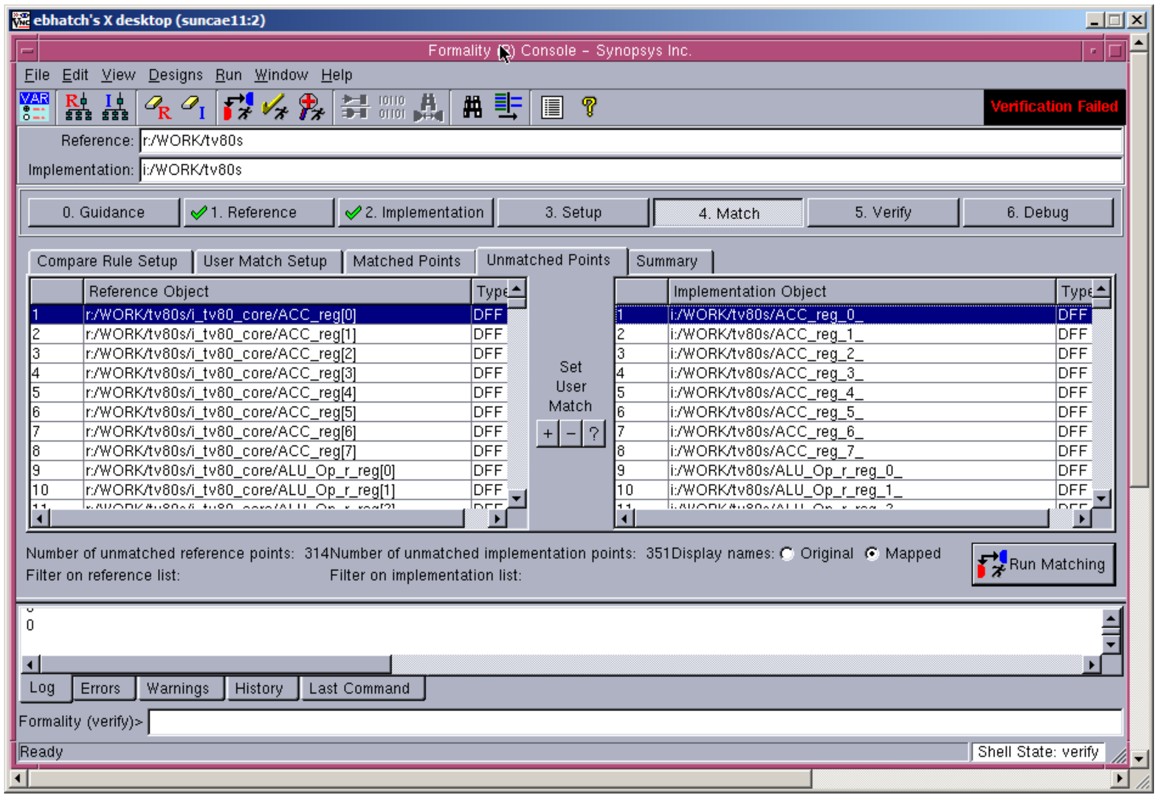
```
    Exists in the impl cone but not in the ref cone for 1 compare point(s):
        r:/WORK/tv80s/i_tv80_core/rfsh_n_reg

-----------
-------------------------------
*************************************************************************************
Analysis Completed
1
```

5.) Messages from matching:

```
******************************** Matching Results ********************************
 77 Compare points matched by name
 0 Compare points matched by signature analysis
 0 Compare points matched by topology
 14 Matched primary inputs, black-box outputs
 314(351) Unmatched reference(implementation) compare points
 0(2) Unmatched reference(implementation) primary inputs, black-box outputs
 177(0) Unmatched reference(implementation) unread points
----------------------------------------------------------------------------------
Unmatched Objects                                                   REF        IMPL
----------------------------------------------------------------------------------
 Input ports (Port)                                                   0           2
 Registers                                                          314         351
   DFF                                                              314         314
   LAT                                                                0          36
   Constant 1                                                         0           1
**********************************************************************************
```

There are several unmatched compare points. This can be one of the problems with this verification.

Here is a picture of the GUI unmatched points tab.



ebhatch's X desktop (suncae11:2)

Formality (R) Console – Synopsys Inc.

File Edit View Designs Run Window Help

Verification Failed

Reference: r:/WORK/tv80s

Implementation: i:/WORK/tv80s

0. Guidance | 1. Reference | 2. Implementation | 3. Setup | 4. Match | 5. Verify | 6. Debug

Compare Rule Setup | User Match Setup | Matched Points | Unmatched Points | Summary

	Reference Object	Type
1	r:/WORK/tv80s/i_tv80_core/ACC_reg[0]	DFF
2	r:/WORK/tv80s/i_tv80_core/ACC_reg[1]	DFF
3	r:/WORK/tv80s/i_tv80_core/ACC_reg[2]	DFF
4	r:/WORK/tv80s/i_tv80_core/ACC_reg[3]	DFF
5	r:/WORK/tv80s/i_tv80_core/ACC_reg[4]	DFF
6	r:/WORK/tv80s/i_tv80_core/ACC_reg[5]	DFF
7	r:/WORK/tv80s/i_tv80_core/ACC_reg[6]	DFF
8	r:/WORK/tv80s/i_tv80_core/ACC_reg[7]	DFF
9	r:/WORK/tv80s/i_tv80_core/ALU_Op_r_reg[0]	DFF
10	r:/WORK/tv80s/i_tv80_core/ALU_Op_r_reg[1]	DFF

Set User Match + – ?

	Implementation Object	Type
1	i:/WORK/tv80s/ACC_reg_0_	DFF
2	i:/WORK/tv80s/ACC_reg_1_	DFF
3	i:/WORK/tv80s/ACC_reg_2_	DFF
4	i:/WORK/tv80s/ACC_reg_3_	DFF
5	i:/WORK/tv80s/ACC_reg_4_	DFF
6	i:/WORK/tv80s/ACC_reg_5_	DFF
7	i:/WORK/tv80s/ACC_reg_6_	DFF
8	i:/WORK/tv80s/ACC_reg_7_	DFF
9	i:/WORK/tv80s/ALU_Op_r_reg_0_	DFF
10	i:/WORK/tv80s/ALU_Op_r_reg_1_	DFF

Number of unmatched reference points: 314 Number of unmatched implementation points: 351 Display names: Original [✓ Mapped]

Filter on reference list: Filter on implementation list:

Run Matching

0

Log | Errors | Warnings | History | Last Command

Formality (verify)>

Ready Shell State: verify

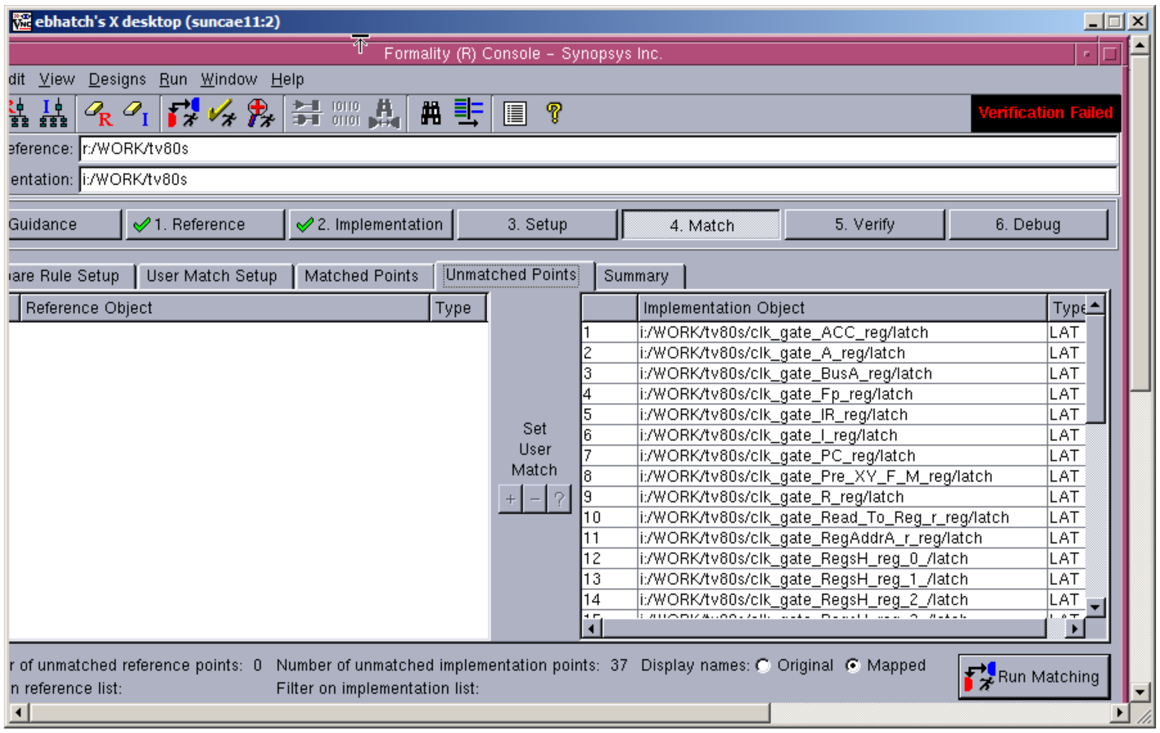
Several compare points look similar but remained unmatched.

6.) At this point we are confident that at least some of the failures are due to matching problems. Try one of the following to correct this issue:

- Use the SVF file.
- Or, write compare rules to fix up matching.

7.) After correcting the script, run through verification again to see what problems remain. Notice that the number of unmatched points has significantly diminished.

8.) View the GUI unmatched points tab again:



eference: r:/WORK/tv80s
entation: i:/WORK/tv80s

Reference Object	Type

	Implementation Object	Type
1	i:/WORK/tv80s/clk_gate_ACC_reg/latch	LAT
2	i:/WORK/tv80s/clk_gate_A_reg/latch	LAT
3	i:/WORK/tv80s/clk_gate_BusA_reg/latch	LAT
4	i:/WORK/tv80s/clk_gate_Fp_reg/latch	LAT
5	i:/WORK/tv80s/clk_gate_IR_reg/latch	LAT
6	i:/WORK/tv80s/clk_gate_I_reg/latch	LAT
7	i:/WORK/tv80s/clk_gate_PC_reg/latch	LAT
8	i:/WORK/tv80s/clk_gate_Pre_XY_F_M_reg/latch	LAT
9	i:/WORK/tv80s/clk_gate_R_reg/latch	LAT
10	i:/WORK/tv80s/clk_gate_Read_To_Reg_r_reg/latch	LAT
11	i:/WORK/tv80s/clk_gate_RegAddrA_r_reg/latch	LAT
12	i:/WORK/tv80s/clk_gate_RegsH_reg_0_/latch	LAT
13	i:/WORK/tv80s/clk_gate_RegsH_reg_1_/latch	LAT
14	i:/WORK/tv80s/clk_gate_RegsH_reg_2_/latch	LAT

r of unmatched reference points: 0 Number of unmatched implementation points: 37 Display names: ( ) Original [✓ Mapped]
n reference list: Filter on implementation list:

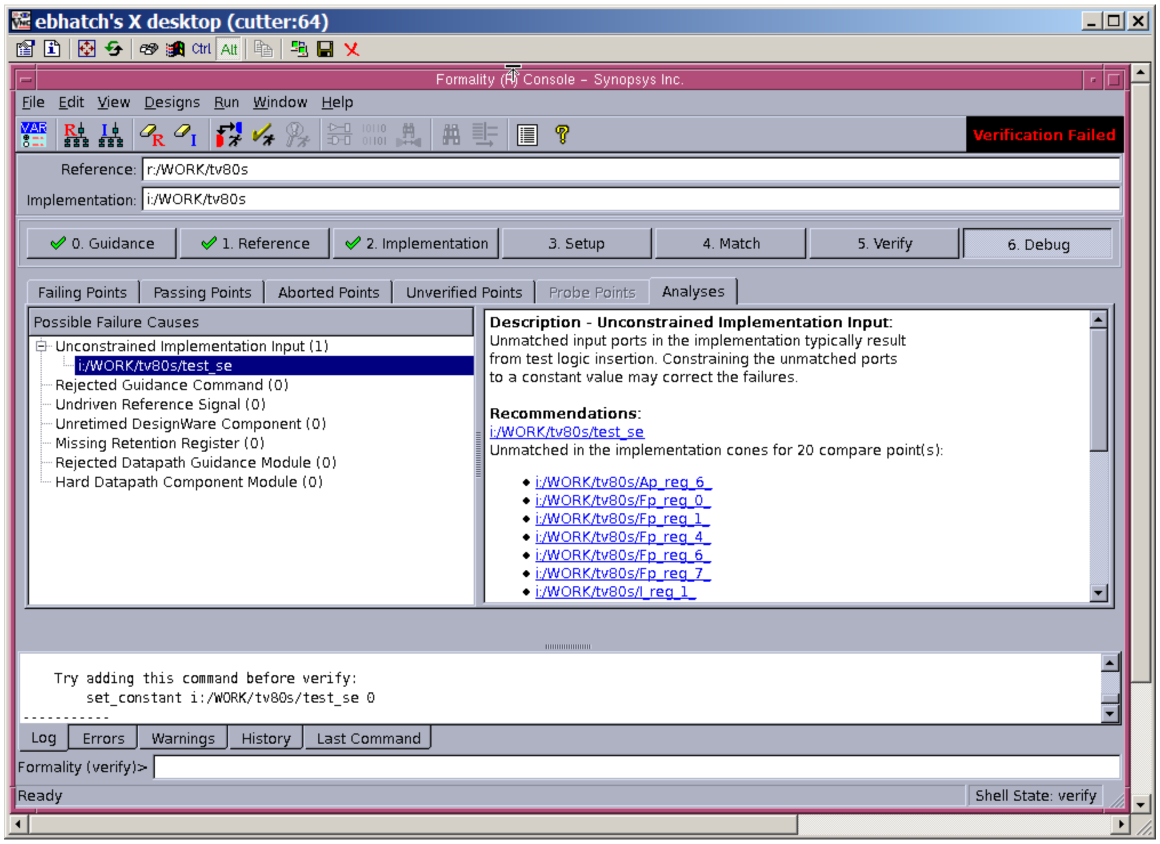
Several of these unmatched objects are "…/clk_gate_..."; however, they are defined as latches. These are probably clock-gating latches.

9.) Resolve the clock-gating latch identification issue by setting this variable:

```
set verification_clock_gate_hold_mode low
```

7.) Re-run verification with the corrected FM Tcl script.

8.) Matching now looks good; however, there are still failing compare points. Try using the "Analyze" feature to get clues for debugging.



Reference: r:/WORK/tv80s

Implementation: i:/WORK/tv80s

Possible Failure Causes

- Unconstrained Implementation Input (1)
  - i:/WORK/tv80s/test_se
- Rejected Guidance Command (0)
- Undriven Reference Signal (0)
- Unretimed DesignWare Component (0)
- Missing Retention Register (0)
- Rejected Datapath Guidance Module (0)
- Hard Datapath Component Module (0)

Description - Unconstrained Implementation Input:
Unmatched input ports in the implementation typically result from test logic insertion. Constraining the unmatched ports to a constant value may correct the failures.

Recommendations:
i:/WORK/tv80s/test_se
Unmatched in the implementation cones for 20 compare point(s):

- i:/WORK/tv80s/Ap_reg_6_
- i:/WORK/tv80s/Fp_reg_0_
- i:/WORK/tv80s/Fp_reg_1_
- i:/WORK/tv80s/Fp_reg_4_
- i:/WORK/tv80s/Fp_reg_6_
- i:/WORK/tv80s/Fp_reg_7_
- i:/WORK/tv80s/I_reg_1_

```
  Try adding this command before verify:
      set_constant i:/WORK/tv80s/test_se 0
-----------
```

Formality (verify)>

9.) Formality indicates that test logic is not disabled, but should be for RTL vs Gate verification.

10.) Set a constant to disable scan mode.

```
setup; set_constant i:/WORK/tv80s/test_se 0 ;verify
```

11.) Fix up the FM TCL script and re-run verification. View the example “runme.fms” under the .solution directory.