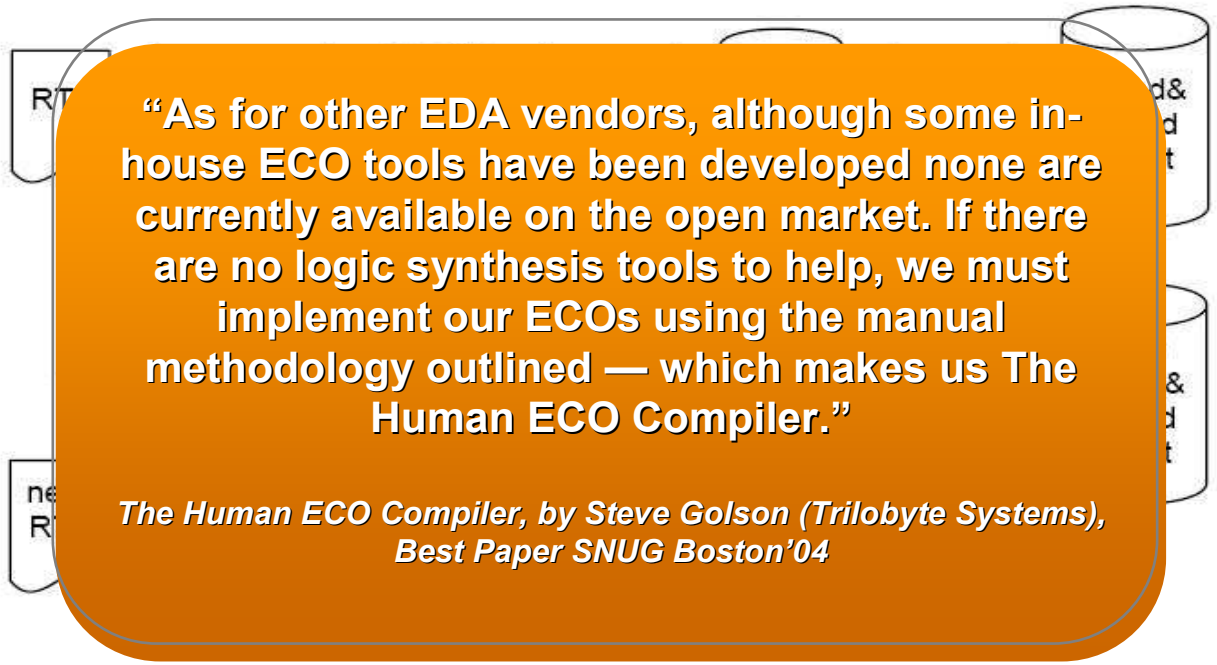
Functional ECO with Conformal Technology

Itai Yarom
Senior CAD Engineer
Design & Verification Specialist
Intel Israel

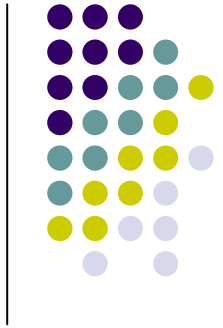
Presented by
Michael Chang
Vice President of R&D
Cadence

What is an ECO?

An ECO is a modification made to an automatically-derived representation of a design. This change is made outside of the normal tool flow.



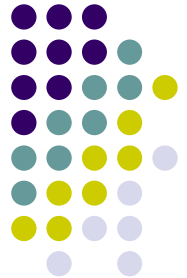
ECO Challenges



- The focus of this presentation is functional ECO's for pre and post TO designs.
  - Non functional ECO's include timing fixes, hold fixes, max capacitance violations, max transition violations and crosstalk problems.
- Functional ECO's are done twice:
  - On the RTL for verification of the ECO correctness.
  - On the netlist, for adding the change in the middle of the implementation model.
- This task is tedious and requires a lot of designer effort.

Why do we need ECO's?

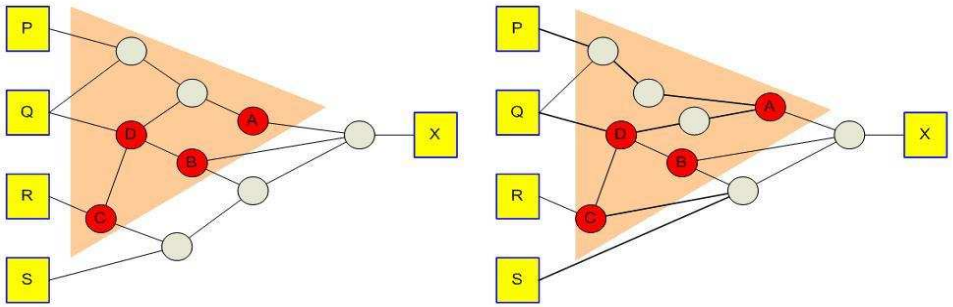
From Steve Golson presentation @ SNUG Boston'04



RTL Frozen →	… but there are a few small bugs
Synthesis is done →	We're not about to run synthesis again, but we're still making changes to the netlist
Placement is done →	We're still running incremental placement on a few ECO cells
Timing closure is done →	…and these three ECOs won't affect it at all
We've taped out the chip →	We've taped out the base layers, and we're still adding metal-only ECOs
This respin is just for timing fixes to improve yield →	…plus 3,600 gates repairing 8 functional bugs

How Formal Equivalence Checking is related to ECO?

1. To compare old RTL to old NL (netlist).
2. To compare new RTL to old NL.
3. To compare new RTL to new NL.
4. To compare old RTL to new RTL.



Logic Equivalence Checking uses formal, static techniques to determine if two versions of a design are functionally equivalent. LEC verify large designs quickly and completely without the use of test vectors.

What are we doing today? What is the challenge?

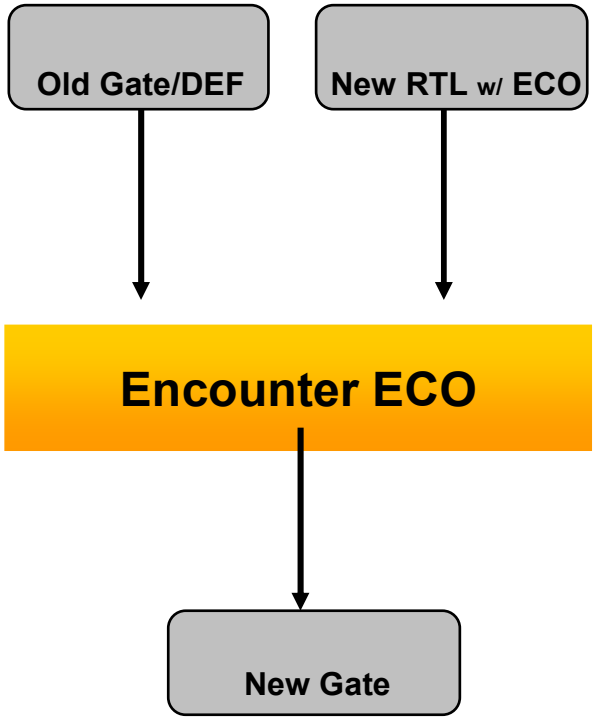
- Functional ECO has 3 steps:
  - Explore the ECO change in the RTL
  - Perform the ECO on the netlist
    - Manually or using Novas Verdi/nECO
  - Fix all the ECO effects in the implementation tools
- What is the challenge in this flow?
- How long it takes to perform each step?
  - Performing the ECO on the netlist can take from several hours to several days (for complex ECO)

ECO Example

- We use for example an ECO that was done in one of our 10 Gigabit Ethernet Controller projects.

- The ECO challenges:
  - Effort: Couple of days, including RTL and Netlist changes, passing LEC and APR fixes due to the ECO.
  - No. of cells: 2394, when the ECO used 367 spare cells.
    - 9 PO's were added and 360 logic cones were effected.
  - Other: Require the most experience engineer to perform the ECO.

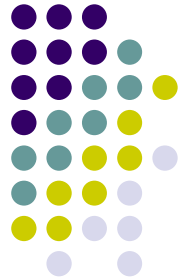
Can it be done differently?



- Provide a complete, automated, and user-friendly *Functional ECO* environment
  - Identify where/what to fix
  - Automatically generate the fix
  - Re-use free gates and spare gates to optimize the fix
  - Support post-mask flow to maximize cost saving through metal layer changes

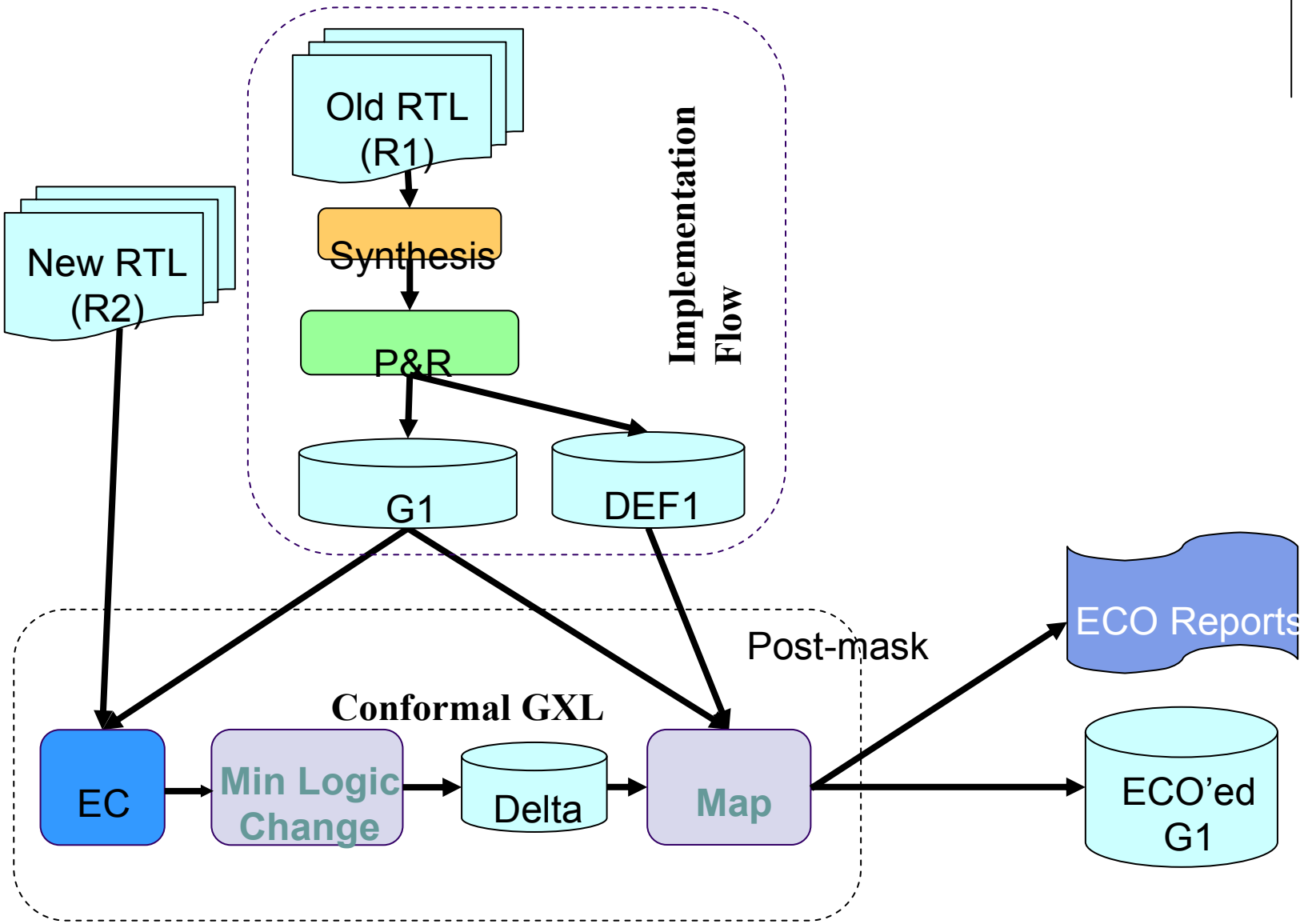
Significantly reduce designer effort and time spent on functional ECO changes

Cadence Conformal ECO

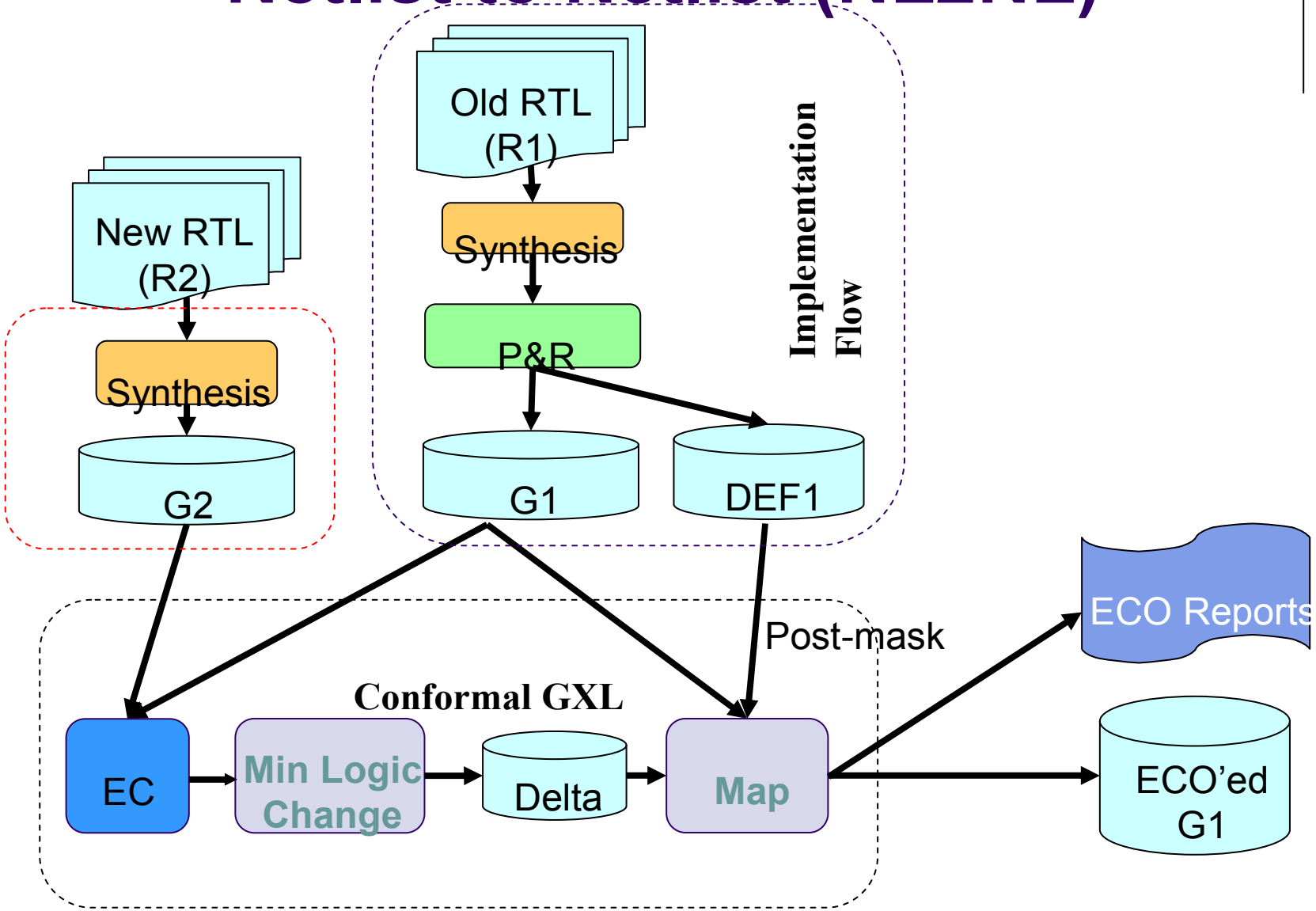


- The Conformal ECO solution offers an automated method to implement functional ECOs.
- The flow has the following steps:
  - Compare pre-ECO to post-ECO files.
  - Create ECO patch
  - Map the ECO patch
  - Write out the ECO netlist
  - Check the ECO results using LEC.

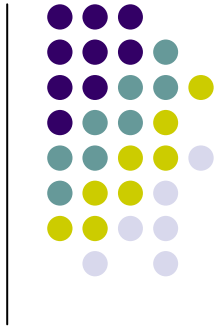
CDS Conformal ECO Flow



CDS Backup flow – Netlist to Netlist (NL2NL)

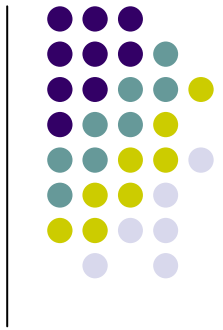


Conformal ECO Report



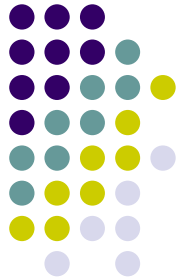
```
//
// Conformal-LEC: Version 06.20-d226 (06-Apr-2007)
//
================================================
            PATCH MODULE STATISTICS
================================================
library cell    : 359
DFF             : 0
DLAT            : 0
primitive       : 66
module instance : 0
================================================
            FREED AFTER PATCH
================================================
library cell    : 450
DFF             : 0
DLAT            : 0
primitive       : 0
module instance : 0
================================================
            RECYCLED
================================================
library cell    : 9
================================================
            NET STATISTICS
================================================
added net       : 84
changed net     : 3
deleted net     : 119
```

Pros & Cons



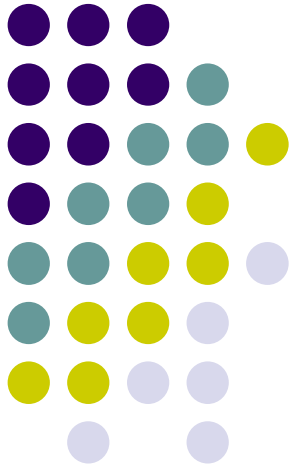
- Pros:
  - Automation of a painful process.
    - Enable to perform more ECO in the same time/resources.
    - Possibility to estimate the ECO effect with a ‘pusg of a button’.
    - Perform the ECO efficiently.
- Cons (& Risks):
  - The tool is on a beta phase
    - We can use the manual ECO flow, in the worse case.
  - The reporting mechanism can be improved.

Summary



- The manual ECO flow that we use today is far from being perfect.
- The conformal ECO provide us a ‘push button’ flow that replace the manual flow.
  - The ECO is guarantied to be correct by construct.
  - In the worse case we can always go back to the manual flow.
- Will the conformal ECO solution will reduce or will it increase the usage of ECO’s?

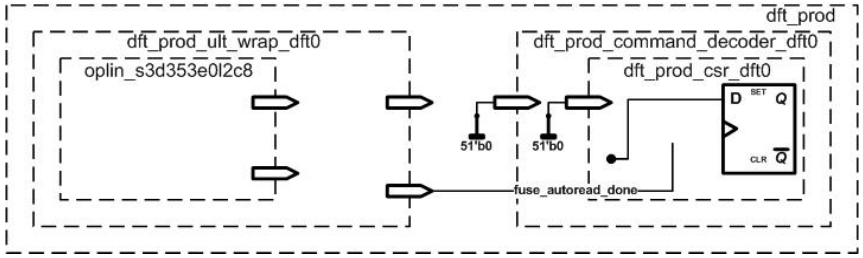
Thank You !



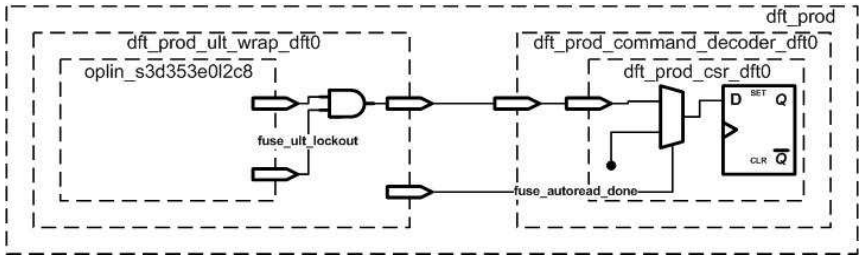
ECO example:
ECO no. 494132 @ Oplin

- Bug Description (2078272):
  - Buses fuse_ult_par_out[51:0] and fuse_mem_par_out[299:0] degenerated to one bit because usage of wrong "logic and" operator instead of "bitwise and"

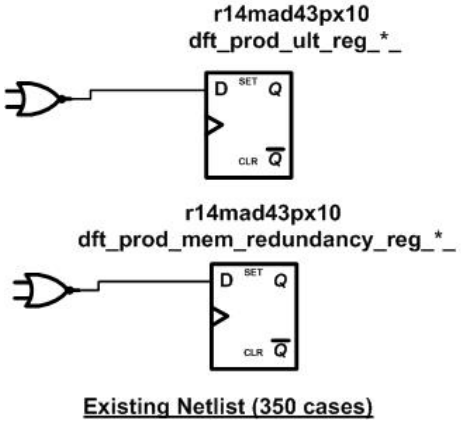
Existing logic

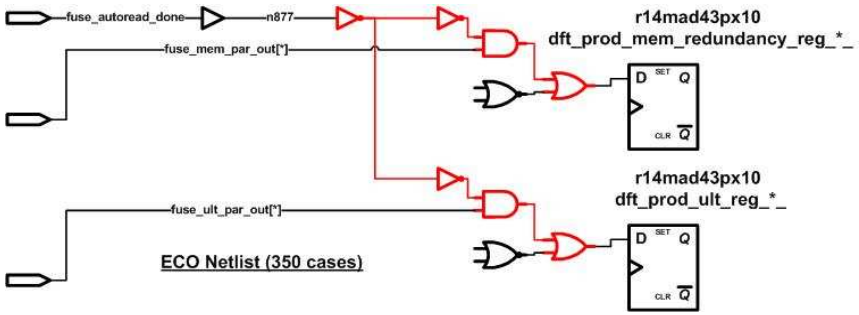


Should be



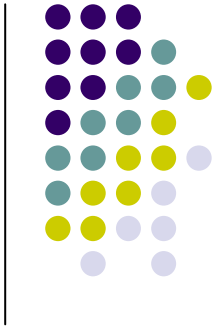
ECO example:
ECO no. 494132 @ Oplin





- ECO challenge:
  - Effort: Couple of days, including RTL and Netlist changes, passing LEC and APR fixes due to the ECO.
  - No. of cells: 2394, when the ECO used 365 spare cells.
  - Other: Require the most experience engineer to perform the ECO.

ECO no. 494132 Details:



Justification:
(why the ECO is needed, what does it fix) :

bug2077284 - atlas_raw_clk is not connected to probe mode bus in core_misc. This signal is not necessary for probe, so RTL will be fixed due to netlist (tie to 1'b0).

bug2077372 - DAT mode - data out [34] not connected to output bin. Data out dft_prod_dat_dout[34] should be connected to port .dat_mode_in_data of pad FLSH_CE_N and pad will be output in dat mode.

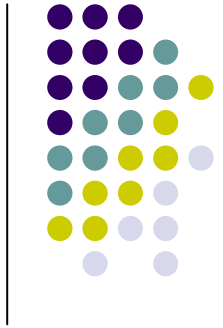
bug2077735 - Probe mode select wrong default value for BI. Defoult value of PROBE_SEL CSR should be output from BurnIn counters - 18'b00_0000_1010_1000_0110

bug2077739 - Wrong direction of 8 i/f pads in scan mode - pads SDP1[2] and SPARE[6:0] should be outputs in scan mode.

bug2078272 - fuse_*_par_out degenerated bus. Misuse of "logic and" instead of "bitwise and" in dft_prod_ult_wrapper cause to fuse_mem_par_out[299:0] and fuse_ult_par_out[51:0] to be degenerated in dft_prod_csr and dft_prod_ult_wrapper. All logic should be repaired. Huge ECO - about 700 cells!!!

bug2078243 - Bits [71:68] of redundency_bus_rx_pb_0 are undriven. 4 MSB were tied to 1'b0 because of wrong assignment width. Demands connectivity fixes.

ECO no. 494132 Details:



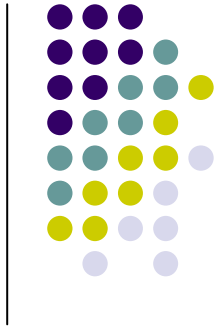
==========
Detailed description:
==========
RTL changes:
(how is it fixed in RTL, Provide clear explanation of the code change before and after):
Fubs changed:

bug2077284 - core_misc.v
bug2077372 - periphery_mux.v
bug2077735 - dft_prod_csr.v
bug2077739 - periphery_mux.v
bug2078272 - dft_prod_ult_wrap.v
bug2078243 - dft_prod.v

Full path of new RTL:
/nfs/site/proj/oplin/oplin/LBDB/rel_oplin/latest/units/dft_prod/logic/src/
/nfs/site/proj/oplin/oplin/LBDB/rel_oplin/latest/units/core_misc/logic/src/
/nfs/site/proj/oplin/oplin/LBDB/rel_oplin/latest/units/periphery/logic/src/

Diff:
See attachment rtl.diff

ECO no. 494132 Details:



==========
Netlist changes (how is it fixed in verilog netlist/sch)
Files changed:

bug2077284 - NA
bug2077372 - periphery_mux
	io_control_block_pfr104
bug2077735 - dft_prod_csr_dft0
bug2077739 - io_control_block_pfr14
	io_control_block_pfr15
	io_control_block_pfr16
	io_control_block_pfr17
	io_control_block_pfr18
	io_control_block_pfr19
	io_control_block_pfr20
	io_control_block_pfr75
bug2078272 - dft_prod
	dft_prod_command_decoder_dft0
	dft_prod_csr_dft0
	dft_prod_ult_wrap_dft0
bug2078243 - dft_prod.v
	dft_prod_command_decoder_dft0
	dft_prod_csr_dft0
	oplin_top

Diff:
See attachment netlist.diff