Formal Verification (Formality) (V1)

Written by: **Fatma Ali**

Content:

==========

What is formal verification:

من اسم verification هنستنتج ان ده check طب على ايه؟؟ هنا انا بقارن بين ال functionality بتاعة RTL code وبين ال netlist الى بطلعها فى خطوات ال flow فممكن تكون ال netlist دى بعد ال synthesis او الى بعد DFT (وده عبارة عن hardware زيادة بحطه فى ال design عشان اعمل testing لل chip اما تتصنع وتيجى وهنتكلم عنه بالتفصيل بعدين بس يعنى نعرف دلوقتى انه اضافة عليا وبالتالى ال netlist بعده هتختلف عن بتاعة ال (synthesis او ال netlist الى اقدر اطلعها بعد placement and routing PNR .... عشان كده بنقول انه بيتعمل post-synthesis & post-DFT & post-PNR (كلمة post يعنى بعد)

يبقى اول مرة هعمله هيكون بعد ال synthesis فانا معايا netlist طلعت من ال synthesis وبنسميها implementation design فمحتاج اقارنها بال RTL code عشان اتاكد ان ال functionality مظبوطة بس ازاى هقارن code مع netlist؟ فانا كده محتاج احول ال code الى netlist بس مش من ال synthesis المرة دى ... وهو ده احد ادوار ال tool الى بنستخدمها فى formality انها بتاخد RTL code تحوله الى gates يعنى مجرد بترجمه الى logic بس وبنطلق عليه reference design وده يختلف عن الى عملته ال synthesis لانها كان عندها constraints لازم تحققها فممكن مثلا تختصر كام gates فى واحدة وباستخدام ال technology library بردوا بتوفر gates مختلفة تخليني احسن ال netlist لكن الى بتعمله ال formality هو مجرد ترجمة اولية للكود بال gates ... وبالتالى ال two netlists هيكونوا مختلفين ك implementation لكن انا عايز ا ال check ال functionality فهل اعمل simulation واشوف ال results؟ ... لا لان ال formality هى **static analysis** مش بتاخد فى اعتبارها كمان اى timing ... فبتعمل ايه؟

بتستخدم Boolean equations & mathematical transformations بحيث تحدد ال functions بتاعة كل حاجة ومن هنا تقدر تشوف هل ال functionality للاتنين متساوية ولا لا ... يعنى كانها بتطلع المعادلات بتاعة كل netlist وتبدأ تقارن بينهم بحيث انهم بيعملوا نفس الحاجة فى الاخر حتى لو ال gates المستخدمه مختلفة

هو ليه اصلا اعمل كل ده مع ان ممكن اخد ال netlist ادخلها على simulator زى ModelSim وادخل معاها ال behavioral library الى اتكلمنا عليها فى Technology libraries file واعمل gate level simulation واشوف ال waveforms قدامى واتاكد من ال output؟ الكلام ده حلو جدا لو عدد ال inputs صغير لكن لو عدد ال inputs كان كبير جدا فمش هقدر انى ا cover كل الاحتمالات الممكنة لل inputs عشان اعملها فى simulation وبالتالى عايزين حاجة تبص على ال functionality من غير اى simulation فهو ده دور ال formality.

EDA tools for formal verification:

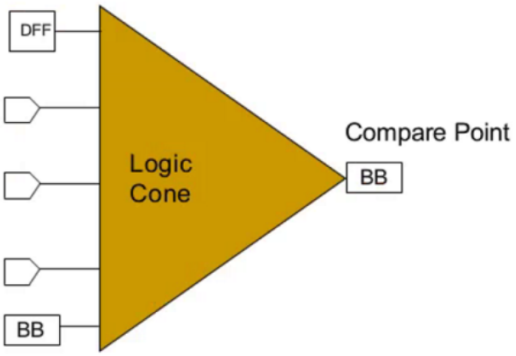
- **Formality from synopsys (more common and we will work here using it)**
- Co-formal Cadence
- Formal pro from mentor

==========

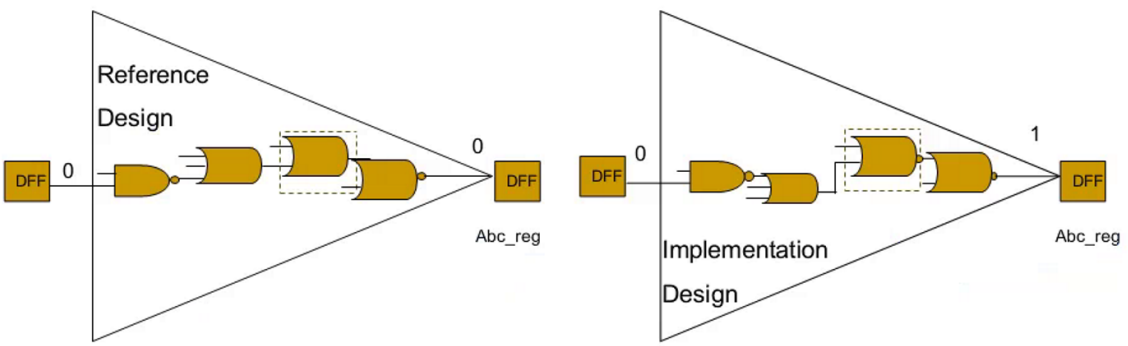
Formal verification component:

ال netlists الى بتطلع وبقارن بينهم دول بقسم ال netlist الواحدة ل two component اساسين:

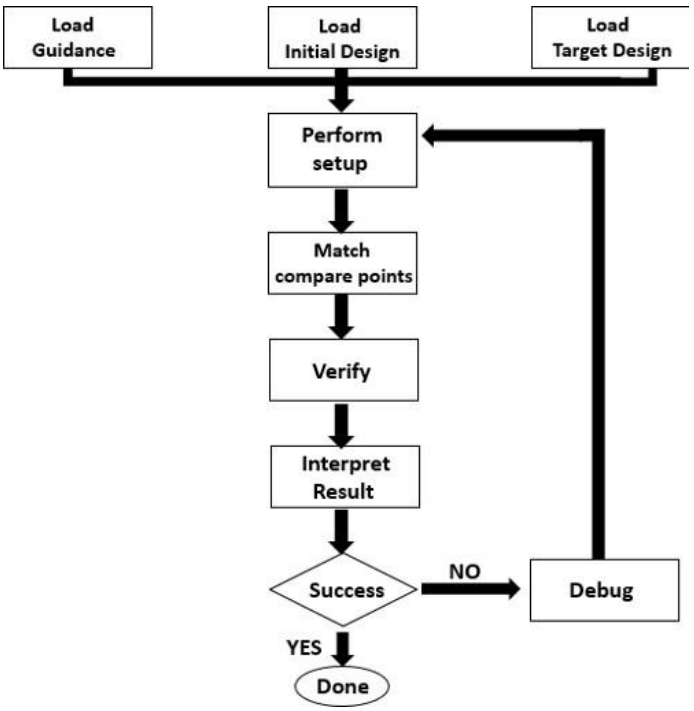
1. اول حاجة **logic cone** وده عبارة عن combinational logic بيتحكم (بي drive) الى بعده والى مش combinational اكيد وبيكون اسمه compare point
2. تانى حاجة هى ال **compare point** ودى الى بعد ال combinational logic على طول وممكن تكون register او latch او primary output يعنى بتاع ال system كله او input pin يعنى بشكل عام دى ال points الى بتبقى output لل combinational logics الى فى ال design فدى هى الى بتطلع معادلاتها من ال two netlist ونقارنهم ببعض



فال netlist بتكون عبارة عن مجموعة من ال logic cones & compare points ... وبعد ما تطلع الحسابات الى قلنا عليها "static analysis" بتقارن ال tool ال logic cones مع بعض لو كان فى اتنين متقابلين مختلفين فى functionality يبقى دول **un-matched logic cone** لكن الى شبه بعض يبقوا **matched logic cones** ففى المثال هنا فى اختلاف gate ادى الى اختلاف فى functionality فبروح اشوف ايه ال errors وارجع اشوف ال synthesis لان مثلا ساعات ال synthesis tool بتلاقيها حطت من نفسها inverter عند output وانت مش عايزه وهكذا فى حاجات فى ال tool ممكن تسبب ان ال netlist from logic synthesis متبقاش نفس المنتظرة ك functionality زى RTL code



Formal verification flow overview:



هنشوف خطوة خطوة تتعمل ازاى بس بشكل العام الاول هدخل ال two netlist عندنا ال implementation design وممكن تسميه target design ثم ندخل ال reference design وبردوا ممكن تسميه initial design (على اساس ان ال RTL موجود الاول فاسمه initial ثم المطلوب كان ال target بقى) وبعدين بدخل ال guidance وده هو(svf.) الى كان بيطلع من ال synthesis وموجود فيه كل اسماء ال signals الى اتغيرت من ال synthesis لان زى ما قولنا فى logic synthesis ان بعض ال signals اسمائها بتتغير زى اى reg signal لو مثلا اسمها "out" هتبقى "out_reg" فعشان اعرف اتعامل مع ال design بعد التغير ده لازم يبقى عندى الاسماء الجديدة دى متسجلة فى file وهو ده ال (svf.) الى بيطلع من ال synthesis وبندخله هنا لل formality ... ثم بعد كده بظبط ال setup وهنشوف يعنى بعمل ايه فيها ... وبعد كده اعمل match بحيث يبدأ يشوف ال compare points الى قصاد بعض ثم verify بحيث يقولى هما متطابقين فى ال two netlists ك functionality ولا فيه مشكلة.

فى طريقتين نقدر نعمل بيهم كده (زى ما خدنا فى ال logic synthesis بردوا) اما انى هكتب script فيه commands بتعبر عن خطوات ال flow ونعمله run مرة واحدة فيفتحلى ال formality tool ويعمل كل حاجة ويقولى فى الاخر verification succeeded ولا فيه errors .... والطريقة التانية هى انى اتعامل مع ال tool ك GUI Graphical User Interface واعمل انا كل خطوة من دول لوحدها وهلاقى بعد كل خطوة بعملها بيتكتب فى window تحت هنشوفها بعد شوية ال commands الى المفروض كنت هكتبها لو مستخدم طريقة ال script هنشوف الطريقتين

اى طريقة هستخدمها فيهم هحتاج افتح ال GUI نفسه او انى هفتح ال terminal اعمل run لل script وهو جواها اصلا command فى الاخر هيفتح ال GUI ... اى حاجة هفتحها هيبقى فى folder هعمله خاص بال formal verification جوا ال project folder بتاعى عشان الموضوع يبقى منظم وكل ال outputs تبقى موجودة ومنظمة فى مكان واحد لكل خطوة فى ال ASIC flow بشكل عام ... فكل خطوة ليها folder خاص بيها

Formal verification flow using TCL script:

خلينا نشوف ال commands بتتكتب ازاى وبعدين نفتح ال terminal ون run

1- Formality Setup:

ده مجرد تظبيط ل mode بحيث تكون synopsys tool مضمون انها صحيت وهتشتغل اهو

Command: set_synopsys_auto_setup true

==========

2- load guidance:

زى ما قلنا انه هنا بندخل لل tool ال .svf بحيث تفهم الاسماء الجديدة الى فى ال implementation netlist دلوقتى وتربطها بأسمائها الحقيقية فى RTL code وتشوف اى تعديل عملته ال synthesis tool … ايه التعديلات؟؟

1. **object name changes** وده الى قلنا عليه ان اسماء بعض ال signals بتتغير وبتتتكب هنا
2. **constant register optimization** وده لما ال synthesis tool بتلاقى reg signal فى ال design قيمتها 0 على طول او 1 على طول بتوصلها ب VDD or GND بدل ما تبقى registered وبده ابقى شلت flip/flop ووفرت فى area
3. **Duplicate and merged registers** لو مثلا عندى two or more registers ولكن بيطلعوا نفس الناتج يبقى نخليهم واحد بس كفاية خارج من ال out المشترك ده

Command: set_svf ”design.svf”

==========

3- load reference design

بنحط ال ref & impl designs فى حاجة بنسميها container وكل واحد فيهم فى container لوحده بنعرفله libraries & design files & top system from the design files فهنا مثلا احنا فى ref design

- load libraries

هنا بحط ال technology libraries بتاعة ال design فى list بحيث احطهم كلهم ولو عندى macros والكلام الى قلنا عليه فى شرح ال libraries وبدخلها بال (.db) عشان كده كاتبين read_db … وبعرف اسم لل container ده الى هو هنا Ref ولو معرفتش اسم هياخد default name وهو " r "

Command: read_db -container Ref [list …..]

====

-Read design files

هنا بحط كل ال HDL code files سواء ال top module او الى جوا كلهم بردوا … وكتبنا عند read كلمة "verilog" لان ال files مكتوبة ب verilog ولكن ممكن نخليها "vhd" لو كانت بكتوبة ب VHDL

Command: read_veriolg -container Ref “Design_file_1”

read_veriolg -container Ref “Design_file_2”

read_veriolg -container Ref “Top_Design_file”

====

- set_top design

هنا بحددله فقط ال top design من الى انا حطيتهم فوق فى read design files بس بحطه على two commands من غير .v فى اخر اسم ال file

Command: set_reference_design Top_Design_file

set_top Top_Design_file

==========

4- load implementation design

نفس الكلام الى عملته لل ref هعمله لل implementation بس هنا بدخل netlist

- load libraries

هنا بحط ال technology libraries وزى ما قلنا بدخلها بدخلها بال (.db) عشان كده كاتبين read_db وهى نفس الى دخلتهم فى ref … وبعرف اسم لل container ده الى هو هنا Ref ولو معرفتش اسم هياخد default name وهو "fr "

Command: read_db -container imp [list …..]

====

-Read design files

هنا بدخل netlist فاى صورة ليها يعنى ممكن ادخل ال .v الى بيطلع من ال synthesis بردوا ولكن هنا اختارنا ندخلها فى formate تانية وهى .ddc واتكلمنا عنهم كلهم فى logic synthesis فالمهم يعنى انى هنا بدخل netlist

Command: read_ddc -container imp “Top_Design_file**.ddc**”

====

- set_top design

هنا بحددله فقط ال top design من الى انا حطيتهم فى read design files فى تعريف ال ref وبحطه على نفس ال two commands وبردوا من غير .v فى اخر اسم ال file

Command: set_implementation_design Top_Design_file

set_top Top_Design_file

5- Some commands (especially for DFT)

ال commands دى بنحتاجها بعد DFT بس خلينا نشوف دورها بشكل عام دلوقتى وفى DFT هنرجع نشوف بقى احتاجنها ازاى

- set_dont_verify_points

هنا انا بدى لل tool بعض ال compare points الى مش عايز اعملها verification اصلا .... وده مثلا هحتاجه فى DFT لان الكود بتاعى فيه ال inputs & outputs & internal signals مضافين عشان ال DFT بعدين لما احطه بالفعل لكن لو احنا بنعمل formality بعد ال synthesis يعنى قبل DFT يبقى هعوز اقوله ان ال ports دى فى الكود متبصش عليها اصلا ولا تعملها verification لانك مش هتلاقى ليها نفس ال functionality فى ال netlist الى طلعت من ال synthesis لانى لسه هحط جزء DFT الى هيحقق ده ... انا مجرد جهزت ال ports & pins الى هحتاجها ليه وبردوا هنفهم الحتة دى اكتر لو فاهمين DFT

Command: set_dont_verify -type port Ref:/work/*/port_name

فى ال type استخدمنا port عشان هنعملها على port الى هى غالبا بتبقى input & output ports الى زادت مع DFT ... ثم عشان احط اسم ال port حطينا اسم ال container الى هو مثلا Ref بعدين " /*/work/" وهنحط اسم ال port بعدها

====

- set_constatnt

هنا انا بدى لل tool بعض ال compare points الى عايز احط عليها قيمة ثابتة يعنى 0or1 .... ودى لنفس السبب بتاع اول واحدة بردوا هحتاجها عشان اعمل test لل functionality لان حسب الى هندرسه فى DFT المفروض هخلى testmode signal = 0 بحيث انه يختار ال path الطبيعى بتاع ال design ومش بتاع ال DFT لان ده الى عايز اعمله formal verification

Command: set_costant Ref:/work/phy_Tx/port_name value

حطينا اسم ال container الى هو مثلا Ref بعدين "/work/" وهنحط ال path الى هيوصلنى لل port

====

- set_verification_set_undriven_signals X

بقول لل tool انها تتعامل مع اى undriven nets هتلاقيها على انها X يعنى don't care بالنسبالى

==========

6- matching & verify

خلصنا كل الى عايزين نعمله لل tool فنقولها بقى يلا تعمل match & verify عشان نشوف الدنيا تمام ولا فى errors

Command: match
verify

==========

7- Debug through formality GUI

عشان لو فيه اى errors اقدر اشوفها واعمل debug من GUI مش هقدر اعمل كده من terminal بعمل فيها run

Command: start_gui

8- Reporting

عشان اتاكد ان الدنيا تمام وكويسة ولو مش كويسة اعرف ايه الى عنده error بالظبط فبطلع شوية reports تقولى ايه ال points الصح وايه الغلط وايه الى unverified بحيث يسهل عليا الوصول لل errors .... (الاسود فى ال commands ده اسم ال report فممكن تحطه اى حاجة عادى)

Command: report_passing_points > “passing_points.rpt”
report_failing_points > “failing_points.rpt”
report_aborted_points > “aborted_points.rpt”
report_unverified_points > “unverified_points.rpt”

==========

8- running

زى ما قلنا هعمل folder لل formality ثم هحط فيه ال TCL script الى عملته وليكن هسميه "fm_script.tcl" ثم من نفس ال file هروح افتح ال terminal (هدوس right click فى ال folder واختار terminal) ثم هكتب جواها ال command الخاص بال formality

Command: fm_shell -f fm_script.tcl | tee fm.log

زى ما خدنا فى ال synthesis ان ال command بالشكل ده بيعمل log file اسمه كده هيكون "fm.log" وفيه كل الى مكتوب فى ال terminal بحيث اقدر ادور على ال errors & warning واى حاجة عامة ولكن عشان اقدر اتعامل مع ال errors واعمل debug هحتاج GUI بردوا فبنفتحها زى ما شوفنا من ال commands

==========

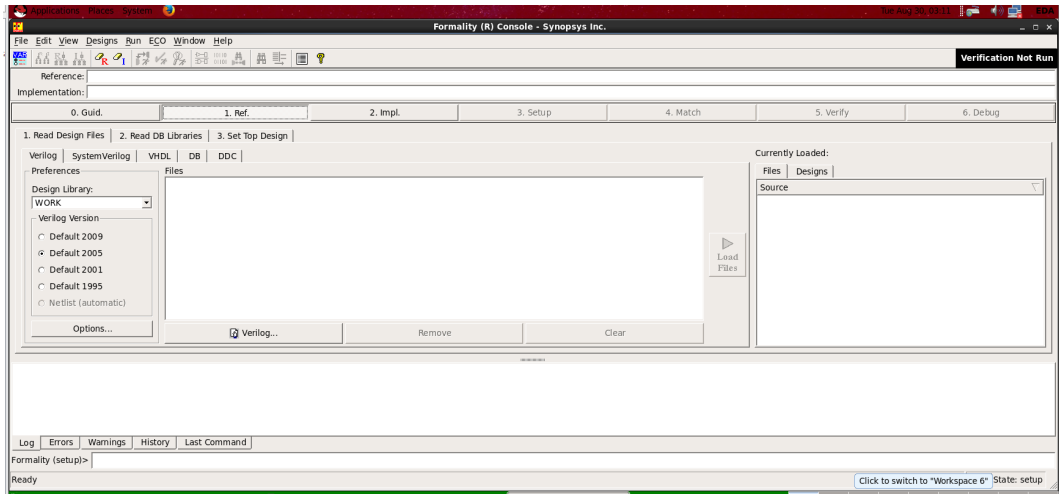
Formal verification flow using GUI only:

هنا بقى احنا هنعمل كل خطوة بايدينا ... هى بالظبط نفس الخطوات الى عملناها بال commands ولكن بنعملها احنا فى GUI نفسه

1- Open GUI

بردوا هنعمل file خاص بال formality بحيث ان كل ال files الى هتطلع تبقى فيه وجواه نفتح ال terminal ونكتب فيها ال command الخاص بفتحه

Command: formality

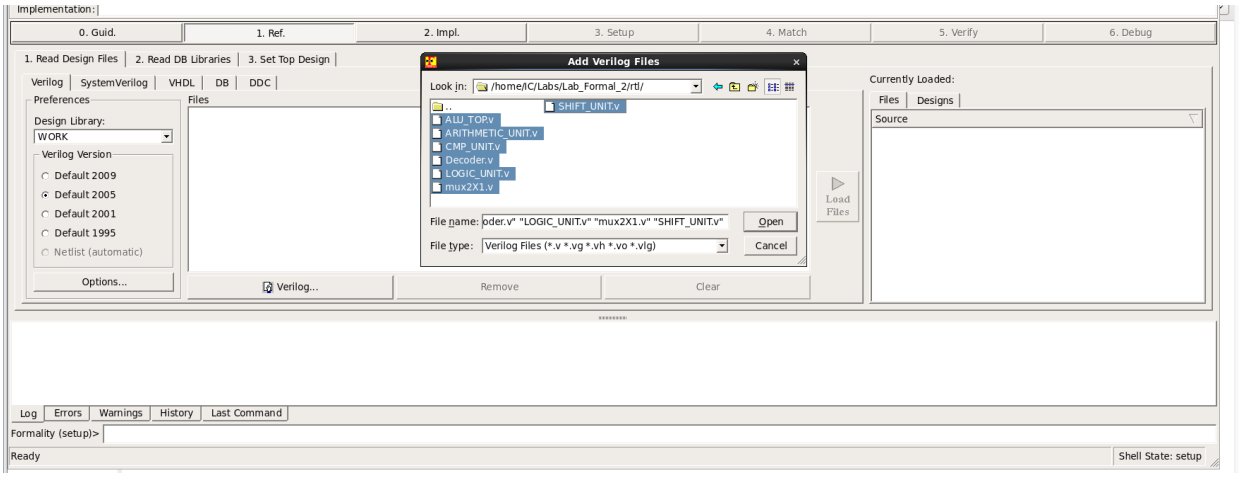


هلاقى عندى كذا tab فى ال GUI ال Guide & set up مش هنستخدمهم هنا .... هنشرح الباقى

2- ref tab

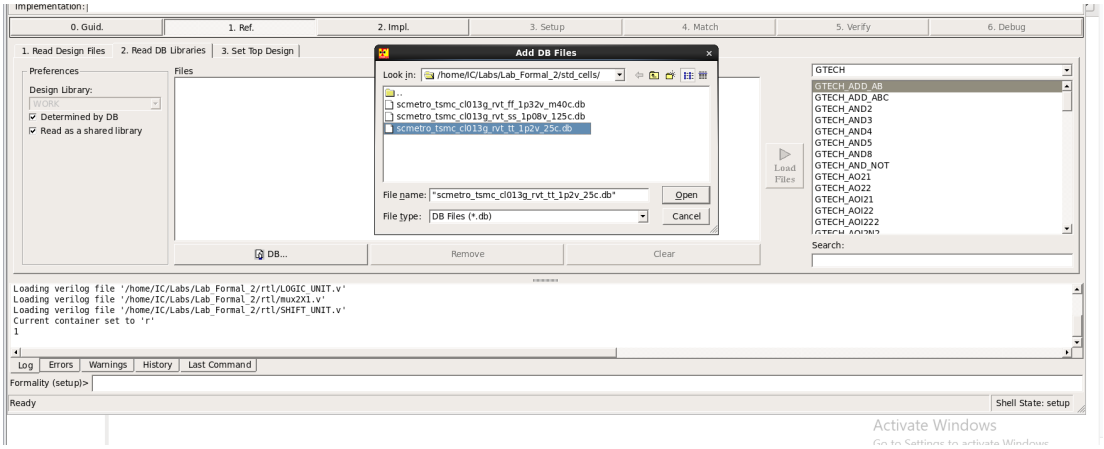
دى جواها اصلا 3 tabs وهنا انا هجهز ال container بتاع ال reference design يعنى ال RTL code وكنا بنعمله ب three commands فهنا هنعملهم على 3 خطوات

1- From Design files tab:

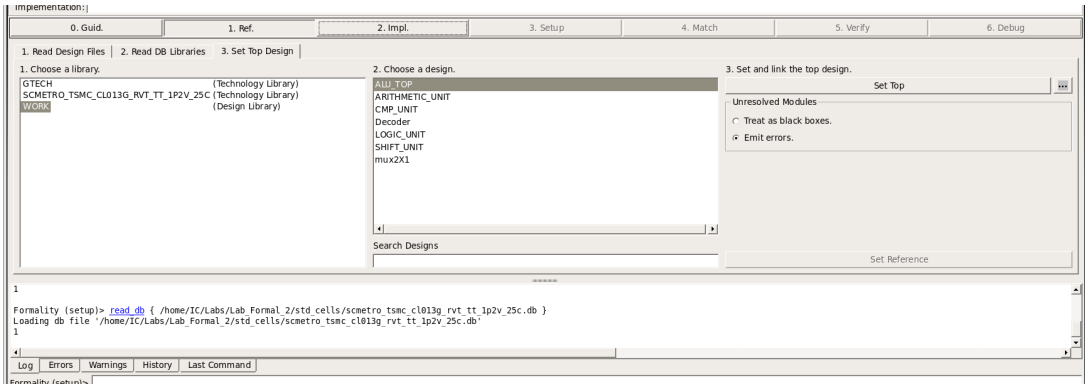


- set the work library and choose the verilog version
- choose verilog tab below and load the RTL code
- Press load design (the gray arrow that will be green)

2- From read DB libraries: choose DB tab below and load your .db library



3- From set top design: choose your top design and click on set top



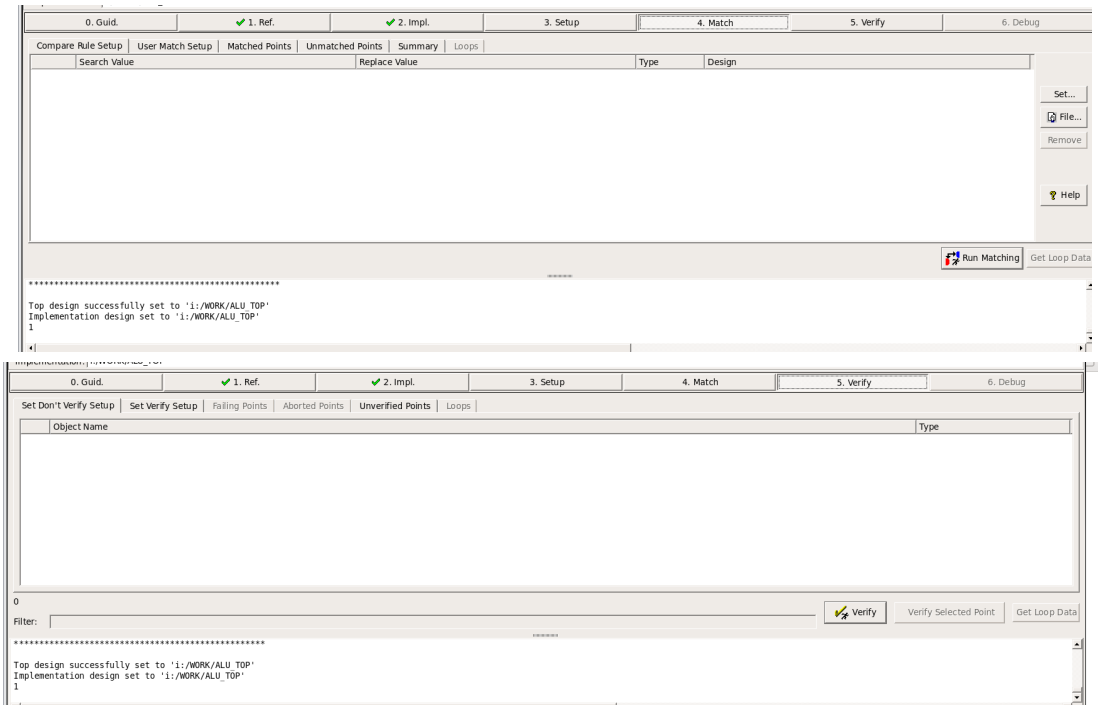
2- imp tab

v. فال netlist يعنى implementation design للcontainer ال ref tab هنعمله هنا بالظبط ولكن هنا بنجهز نفس الى اتعمل فى الى هنختاره هنا فى ال design file هو netlist مش code

==========

3- match & verify tabs

هندخل ال match الاول ونختار run match ثم لو تمام نروح ل verify tab وندوس على verify



==========

4- failed or succeeded verification

لو succeeded هلاقى كلمة verification not run فوق بقت verification succeeded وتمام لكن لو فى غلط هيطلعلى verification failed ويودينى عند ال debug tab

==========

Some GUI Features:

وحتى لو استخدمت TCL script بتفتح فى الاخر GUI تشوف عليه بس فهتكون الحاجات دى مفيدة بردوا:

1- ممكن تشوف ال schematic بتاعة ref or imp من icon اسمها view design Architecture ثم من جوا تختار ال design انى واحد ثم from view menu choose view design ---- ده شكل ال icon

2- ممكن نحفظ شكل ال schematic الى طلعت من (1) عن طريق:

schematic menu then choose print and save it as PDF

3- ممكن من ال architecture الى طلع فى (1) بردوا تشوف ال source بتاعه سواء كان RTL code لل reference design او كان netlist لل implementation design من:

view menu then choose view source

OR Right click on schematic then choose view then view source

==========

Contact info:

Linkedin profile: linkedin.com/in/fatma-ali-57b1a6200

E-mail: fatma.ali.2028@gmail.com

All ASIC files are provided on VLSI - ASU Community:

https://drive.google.com/drive/folders/1aLwCLZj0YG9KJhln1D60_nWM7p-L3q9a?hl=ar

==========