Automated Determination of Top Level Control Signals

Rohit Kumar Jain Praveen Tiwari Soumen Ghosh
Synopsys Inc.
Bangalore, India
{rkjain,ptiwari,gsoumen}@synopsys.com

***Abstract*— During various stages of hardware design, different types of control signals get introduced; clock, reset are specified and connected at the RTL stage whereas signals like scan enable, isolation enable, power switch enable get added to implemented devices later in the flow.**

The quality of Top Level Control Signals (TLCS) has a direct impact on the quality of static verification which is used to verify the intended connectivity and functionality of fan-out networks corresponding to TLCS. Typically, users need to specify these TLCS (along with their intended types) for such static verification. But when TLCS are not known to the verification engineer, reverse-engineering of clock, reset and scan network implemented in a design becomes a non-trivial task.

This paper proposes a framework to automatically generate a list of TLCS pertaining to the implemented design. The framework describes a heuristic-based analysis of fan-in cones, traversing backwards from the leaf cell instance pins. It is independent of design style(s) as its core strength lies in its capability to dynamically adapt to the new discoveries of the design elements made during the traversal.

Keywords— Inference of Top level control signals, Static Verification, Low Power

I. Introduction

Verification of an implemented design is an integral part of the design cycle before it can be signed off to the foundry for final fabrication. Verification is becoming a huge bottleneck with each passing day due to the increasing size and complexity of the design as well as usage of advanced low power techniques. Employing static verification techniques at various stages of the design can prevent a lot of design bugs at an early stage thereby reducing the overall cost impact. In order to perform such static checks there is a need to find out signals that are indispensable to correct design behavior [4]. The framework for coming up with the truly important TLCS affecting clock, reset, scan, power enable, etc. is explained in the later section(s).

A mechanism to derive the intended TLCS from the implemented design would greatly aid the user in verification. This is not so much of a problem at RTL stage where designers are aware of the design elements; it's more at the post synthesis, routing stages that specification of such signals poses a challenge because of the complex power gating, clock gating and other such methodologies used. Tools do provide mechanisms such as 'tracing' which verification engineers can employ iteratively to traverse the design. Such an analysis starts by performing fanin traversal from specific leaf cell pins (example: clock pins) all the way to top level/generated signals. As paths can contain sequential, complex logic, the responsibility of constraining the traversal falls on the user. Such approaches are referred to as naïve approaches in the rest of the paper. A naïve approach to determine the TLCS could end up identifying false, wrong type of TLCS (for example: scan enable TLCS getting identified as a clock signal).There are other proposed methods which can be used for simulation frameworks [3] .However, those methods are mostly restricted to only a few kinds of signal(s).

The proposed framework allows for traversal through most combinational and sequential elements (like flip flops and latches) as well as complex combination clouds. It takes into account structural attributes and topology of the design nodes (for example: the determination of the reset TLCS takes into account the design nodes encountered, hierarchy levels traversed, overlay with scan networks etc.) The framework is very useful where 1) the design complexity is very large and ends up in a large number of TLCS or 2) the TLCS generating logic is very complex and requires a tool for its determination.

The framework not only comes up with a concise set of such signals but also provides guidance to the user for cases where the determination fails. The results of this approach on a variety of designs, of varying sizes and complexities, show a near complete list of extracted control signals, vis-à-vis manual specification.

II. Background

The verification of the micro-architecture of the system-on-chip (SoC) is becoming increasingly challenging with the increasing SoC complexity. In order to ensure the performance and power requirements, designers should be able to capture the entire system at a high level of abstraction early on in the design cycle [6].

In a complex SoC, TLCS that the verification engineer intends to verify often pass through cone of logic where it interplays with other control signals or data and finally reaches the desired cell instances. A simple example, as shown in Fig. 1, shows the clock (CLK), test clock (TCLK) and test enable (TE) signals going through logic cloud which selects either clock or test clock depending upon the value of test enable signal and finally reaches the clock pins of flops (or any clocked sequential logic) in the design. The test enable signal also reaches scan enable pins of the same set of flops in the design. A naïve approach to extract clock signals would be to potentially infer all control signals clock, test clock and test enable as clock control signal and test enable will also get inferred as scan enable control signal.

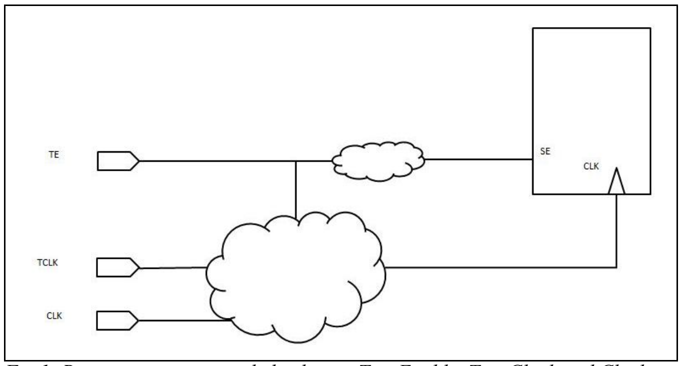

Fig.1: Representative network displaying Test Enable, Test Clock and Clock

Similarly, for the scenario as shown in in Fig. 2, the naïve approaches infer four clock TLCS as has been marked with the arrows. However, there is only one top level clock source in the design – rest all is noise from verification point of view. These extra elements get introduced at various stages of implementation such as power/ clock gating logic insertion, testability structure insertion etc. The problem of these inferred signals gets even worse if there is interplay between these control signals. Even though the above examples have been explained given using the clock signals which can be specified using the design constraints format [5], the problems highlighted are applicable to other types of TLCS such as reset, isolation enable, etc. as well.

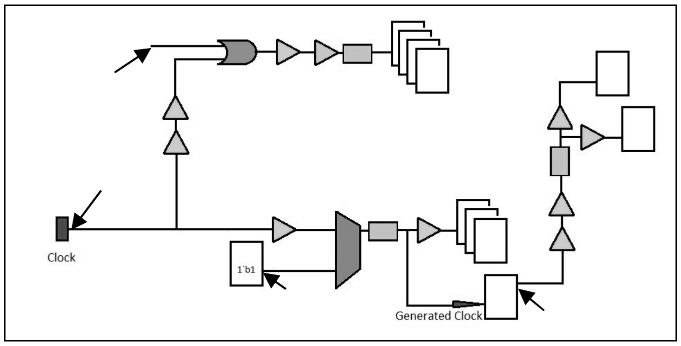

Fig. 2: Sample Clock Network where naïve approaches infer wrong TLCS

The existing design constraint formats either don't support some of these types or for other types that definition of TLCS in terms of design constraint is not available to the verification engineer who would generally belong to a different design group. If the TLCS, inferred by a naïve approach, are directly provided in the verification process; a static verification tool may either miss the real design bugs or create noise in the report due to poor quality of the TLCS inferred. The effort to filter such noise increases the cost of verification. Such costs can be reduced if the TLCS reported by the naïve approaches were filtered before passing it to the static verification tools. But unless this filtration process is inbuilt in the inference step itself, the filtering mechanism either will be error prone or resource (time or people) intensive

III. Generic Framework

To aid the filtering process, we introduce a coloring approach in the inference step.

In this process, a distinct color is selected for each type of TLCS that the user wants the tool to infer. Then, based on the attributes in the design, the starting nodes are identified for the inference of TLCS. These starting nodes are assigned with a pure color depending upon its type. The fan-in cone analysis is then started for each of the starting point selected as per the attributes. Any design node that is encountered during fan-in analysis gets painted with a transformed color. A transformed color is one of a pure color, a reduced concentration of a pure color or mixture of reduced concentration of two or more pure colors. As the fan-in analysis crosses logic, the concentration of transformed color is reduced by the cost of the logic. Eventually, all end points of fan-in are marked with transformed colors.

After the fan-in cone analysis, groups of all end points marked with transformed colors are formed. The grouping can be of two types - pure color groups and mixed color groups. Members of pure color group have only one constituent pure color. Each member of mixed color group has two or more constituent pure colors with reduced concentration. It is possible for members of pure color groups to have varying concentration of the pure color. Every pure color group can be divided into one or more sub groups depending upon the distribution of concentration range. Similarly, a mixed color group can be split into multiple sub groups based on prominent color for each of its members. Further classification of each of these sub groups formed out of a mixed color group can be done with distribution range of prominent color's concentration. Finally, some acceptance criteria is employed – the acceptance criteria could be either inherent in the tool or user specified - to select or deselect members of final subgroups classified in terms of the concentration distribution of pure color.

IV. Case Study

This section explains an example implementation of the generic framework explained in section III, where the coloring with only the pure color approach is explained in terms of the weights.

Initially all the control signals are identified on the basis of the attributes - namely design or the power format attributes. While traversing back from these control signals all the traversals are given an initial base weight of 1.0. For a very simple example case of traversal through a 2-input equiprobable AND gate, the static probability of the control signal being affected by the value of the input if it passes through the gate is 0.75. Hence, once a traversal is done through the AND gate, we reduce the weight of that traversal by a factor of 0.75. All the weights are listed in Table 1.

Similarly, the reason for diminishing the weights significantly when the fan-in traversal starts analyzing flop pins where the pin under consideration is other than the data pin, is that the probability of that signal being the top level control signal is close to zero. For example: An isolation enable top level control signal fanin traversal hits a save pin of a flip flop, then there is a very low probability of that save pin being the control signal for the isolation enable. Hence, it has been diminished to a great extent. The data pin however depends only on whether the flop is in enable mode and whether the clock is in ON state, thereby contributing to the weight. Other weights can be explained similarly.

TABLE I. WEIGHT FACTOR BASED UPON CELL TYPES

Cell Type	**Weight factor**
AND,OR,NAND,NOR	0.75
BUFFER	1.0
FLIP FLOP DATA PIN	0.25
FLIP FLOP CLOCK PIN	0.001
FLIP FLOP SAVE PIN	0.001
FLIP FLOP RESTORE PIN	0.001
FLIP FLOP CLOCK TEST PIN	0.0001
FLIP FLOP SCAN CLOCK PIN	0.00001
LATCH	0.25
TRISTATE	0.0625
OTHER COMBINATORIAL	0.0625

The function also identifies the hierarchy level reached by the fanin traversal. If a traversal reached the directly user controllable pin, this would imply that this signal will be user controllable and hence it could directly manipulate the value of aforementioned control signals, thus there is a need to augment the weight for that specific traversal. The number of endpoints for the traversal are in general very large, hence ensuring that the statistical distribution for a given fan-in traversal would be a normal distribution. Thus for a set of traversal endpoints, their hierarchy levels are computed and their corresponding hierarchy level mean μ_{hier} and variance σ^2_{hier} are calculated and then the augmentation of weights (W_i) is done on the basis of the 75% acceptance criteria of the normal distribution [8] as in Eq. (1):

$$W_i = \begin{cases} 2 \times W_i (if \;\; abs(W_i - \mu_{hier}) < \;\; 0.674\sigma_{hier}) \\ 5 \times W_i (if \;\; W_i - \mu_{hier} < \;\; - \;\; 0.674\sigma_{hier}) \\ W_i \; otherwise \end{cases} \quad (1)$$

These new weights for a particular control signal with mean (μ_w) and variance (σ^2_w) are huge in number and the central limit theorem would imply that this distribution would tend toward being a normal distribution of weights. All the weights which are greater than the 75% acceptance criteria [8] of the normal distribution are accepted as in Eq.(2) based on the boolean value of the $Flag_i$ variable:

$$Flag_i = \begin{cases} True \, (if \;\; (W_i - \mu_w) > - \; 0.674\sigma_w) \\ False \; otherwise \end{cases} \quad (2)$$

The $Flag_i$ value denotes whether the control signal under analysis will be reported or not to the user. Even the above heuristic was not able to "filter" out unrecognized cells such as blackboxes.

To aid the filtering of complex cells over and above the previous heuristic assignment of negative weights was also tried. However, just the negative weights assignment is not a feasible solution for cases where such unidentified cells are huge in number. In such cases, the mean would be highly biased towards these negative weights and these signals would invariably figure in the "filtered" set of TLCS. Thus, most of these negative weighted fanin endpoints would figure in the final TLCS, so the implementation removes such fanin endpoints from the variance computation itself. The removal allows for the variance to be "unbiased" towards these negative weighted fanin endpoints thereby ensuring that the methodology filters out these complex cell endpoints as well as other fanin endpoints which are out of the acceptance region during the new variance computation.

TABLE II. WEIGHT FACTOR FOR SPECIAL CELL TYPES

Cell Type	**Assigned Weight**
MEMORY	-2.0
BLACKBOX INSTANCE	-2.0
UNIDENTIFIED SEQUENTIALS	-2.0
UNIDENTIFIED CELLS WITH NO GATE TYPE	-2.0

The implementation over and above all the discussed filtering mechanism identifies the intermediate unconnected and constant endpoints and marks and filters them. Even with the above explained heuristics, the implementation does not hide any TLCS from the user. It just filters them out and allows the user to provide the inputs and choose the signal if the user wishes. The whole process is explained in a flow-chart as shown in Fig. 3.

Naïve approaches would infer all the endpoints as the fanin traversal to be TLCS, as shown in Fig.2. However the implementation "filters" out the intermediate unconnected and constant tied endpoints based on the weightages it would have accumulated in the traversal. Although in one fanin traversal the hanging endpoint and the "real" clock would have same weightages, the implementation has the knowledge to filter out the intermediate unconnected one.

The significance of the reduced TLCS is reflected from the fact that when these were fed to a low power connectivity check [7], the results were the same as it would have been with the enlarged set of TLCS which were inferred with the naïve approaches, thus proving the efficacy of the method and reducing the debug time for the verification engineer.

V. Flow Chart

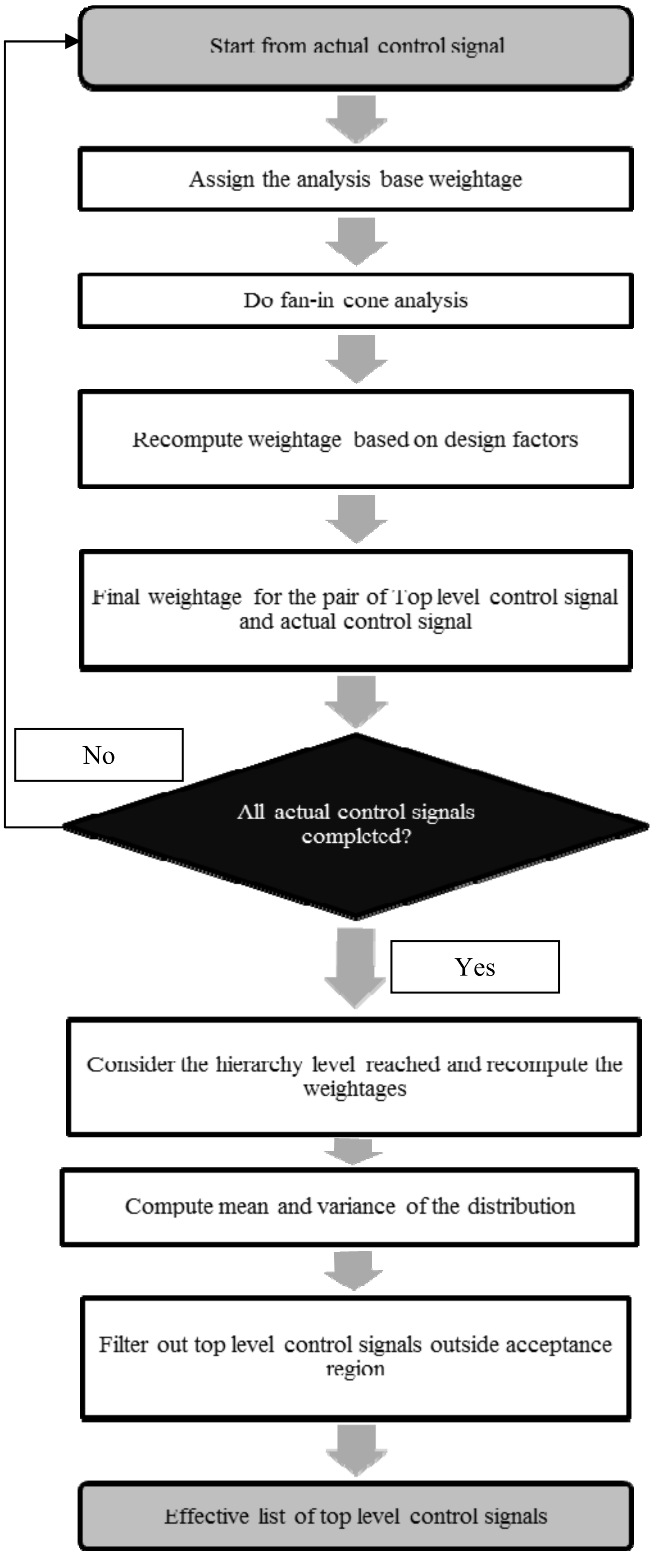
Start from actual control signal
Assign the analysis base weightage
Do fan-in cone analysis
Recompute weightage based on design factors
Final weightage for the pair of Top level control signal and actual control signal
No
All actual control signals completed?
Yes
Consider the hierarchy level reached and recompute the weightages
Compute mean and variance of the distribution
Filter out top level control signals outside acceptance region
Effective list of top level control signals

Fig. 3: Flowchart of the process followed

VI. Results

The results clearly show that the proposed methodology is able to reduce the number of inferred TLCS in the designs. All the designs in these results are actual customer design with more than 100000 instance.

TABLE III. Results

BENCH -MARK	LEAF INST	TLCS I	TLCS II	TLCS	LOW POWER VIOLATIONS
A	594601	3338	2333	66	3
B	369950	1560	2060	145	229
C	192793	19186	18925	728	52
D	441970	5236	5183	1193	20
E	1231467	13436	14285	3054	357
F	7199004	213383	175627	161057	8718
G	28031423	141313	188490	54962	7981

In the above results, for the designs with the mentioned number of module leaf instance (leaf inst), TLCS-I and TLCS-II have been obtained by using two different naïve approaches whereas TLCS are the ones obtained from the example implementation. As mentioned in section four, the number of violations is same even with the lower number of TLCS obtained with the implementation.

VII. Future Work

The current implementation only considers the pure coloring approach explained in the previous section(s). The observation, however, has been that the coloring could be of the transformed nature as well. There is a scope where the filtering needs to take into account the transformed coloring as well to reduce the number of TLCS and in turn reduce the debug time for the verification engineer even more.

References

[1] Michael Keating , David Flynn , Rob Aitken , Alan Gibbons , Kaijian Shi, Low Power Methodology Manual: For System-on-Chip Design, Springer Publishing Company, Incorporated, 2007

[2] Saint-Preux F., MVRC Usage on a Complex Design from the RTL to the Low Power Signoff, SNUG (France, 2010).

[3] J.-P. Talpin and S. K. Shukla. 2005. Automated clock inference for stream function-based system level specifications. In *Proceedings of the High-Level Design Validation and Test Workshop, 2005. on Tenth IEEE International* (HLDVT '05). IEEE Computer Society, Washington, DC, USA, 63-70. DOI=10.1109/HLDVT.2005.1568815

[4] A. Goel and W. R. Lee. Formal verification of an ibm coreconnect processor local bus arbiter core. *37th Design Automation Conference*, June 2000.

[5] Synopsys Design Constraints http://www.synopsys.com/community/interoperability/pages/tapinsdc.aspx

[6] Y. Mathys and A. Châtelain, “Verification strategy for integration 3G baseband SoC,” in *DAC ’03: Proc. of the 40th conference on Design automation*. New York, NY, USA: ACM Press, 2003, pp. 7–10.

[7] Synopsys (2008) Low-Power Flow User Guide V-B-2008.06

[8] Standard Deviation for Normal Distributions http://en.wikipedia.org/w/index.php?title=Standard_deviation&oldid=527317893