International Journal of Electronics and Telecommunication Systems
Volume1, Issue 03 2025

Use of Abstraction for Generating Timing Models for Hierarchical Design Closure

Rajagopal Nilava[1], K.K. Loo[2]
Faculty of Electrical Engineering and Electronics at Technical University of Gabrovo

Research Article

Date of Submission: 15-05-2025 Date of Acceptance: 19-06-2025 Date of Publication: 20-08-2025

ABSTRACT: Today's designs need hierarchical timing analysis because of the physical limitations of hardware and software, as well as the complexity and scale of these systems. One crucial technique that enables the lightweight representation of design blocks for timing closure is timing model generation. In this work, we examine these models from the standpoint of accuracy and make predictions about future directions that might enhance them for hierarchical timing closure.

KEYWORDS: Timing, Closure, Extracted, Interface, Model

INTRODUCTION

Timing closure has proven to be very difficult as design sizes and complexity have increased. In several phases of application-specific integrated circuit (ASIC) development and sign-off procedures, static timing analysis (STA) [1] has emerged as a key step that is employed iteratively. The main goal for ASIC flows is to choose a corner that is characterised by a certain process, voltage, and temperature, then aim for timing closure on that corner. A predetermined set of corners and modes are identified on which the timing analysis is performed to ensure that the functionality is intact because traditional methods of corner sign-off at extreme corners do not really guarantee that the targeted chip will work due to the decreasing design sizes and increased variation.
Additionally, STA must be performed for each constraint mode if the chip is designed to operate in many modes. This raises the difficulty of executing STA on several corners and functional mode combinations, which has a big impact on the chip's time to market.
This need led to the discovery of other ways to lower the number of STA runs required to properly declare that the chip's intended functionality is accurate. Using timing models [2] for a certain hierarchical block of design is one of such options. This work focuses on library-based timing model creation, which is accomplished via the abstraction of block timing. There are many ways to generate timing models.

BRARY BASED TIMING MODEL GENERATION

Timing information for the pathways of interest is abstracted as timing arcs of a modelled timing library that may ultimately be recorded onto disc in library-based timing model development.

LIMITATIONS OF LIBRARY BASED TIMING MODEL GENERATION

Even while these models can accurately depict the temporal features of interface logic up to a certain point, they nevertheless have a number of drawbacks that are necessary for today's designs. We have attempted to enumerate the most obvious shortcomings that may be fixed by including some more information into the library.

First and foremost, the situation when many produced clocks [3] are applied at a certain time in the design is a restriction that makes the whole library-based timing model creation pointless. The issue arises when using such a model since there is no abstraction of master clock information relevant to these

produced clocks, even if one tries to abstract this information within the timing model.

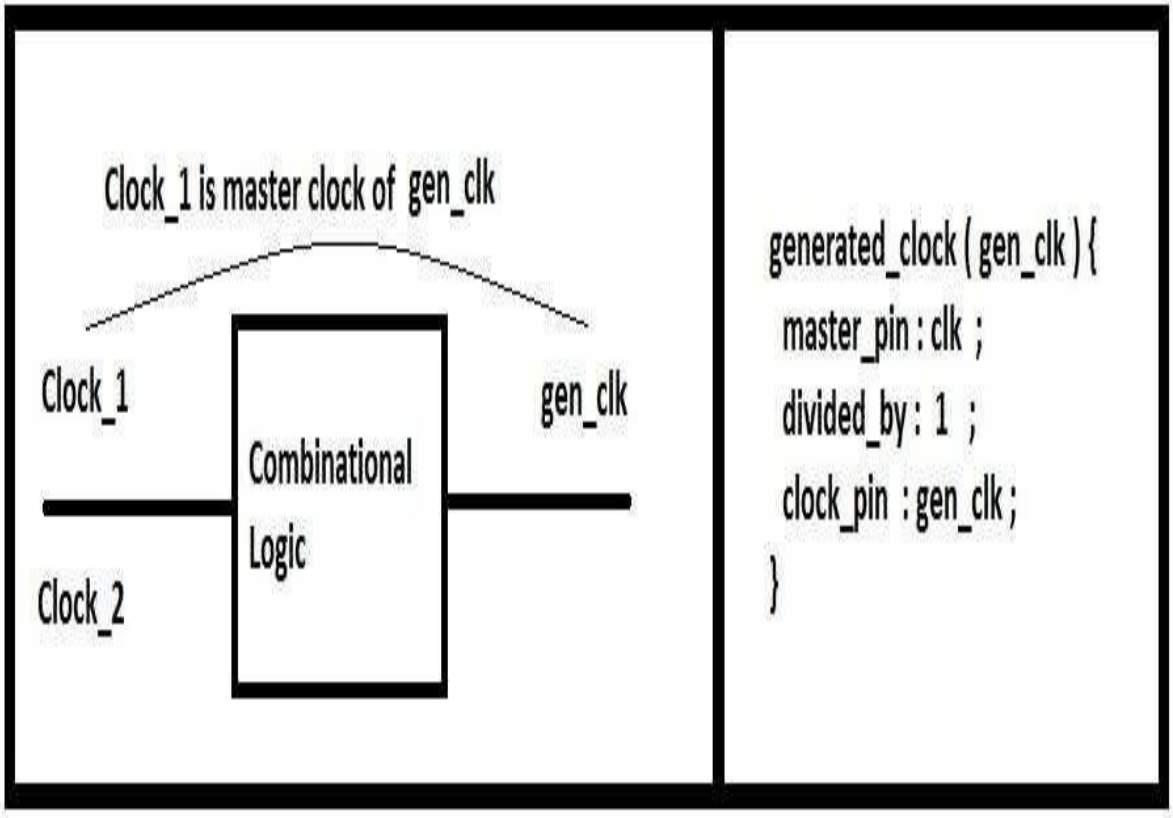
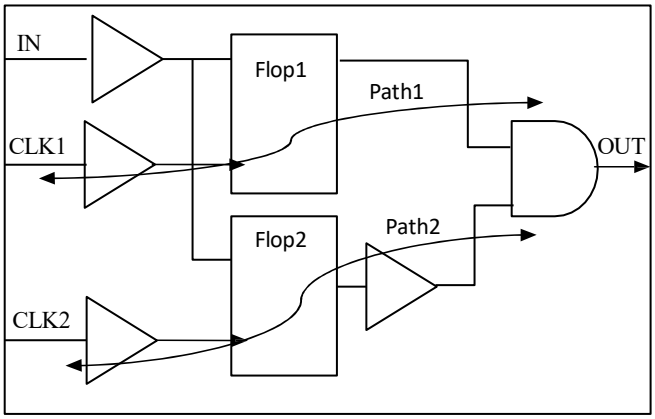


Fig. 1

To illustrate above point, let us consider the above circuit. We have to clocks “Clock_1” and “Clock_2” defined on a port named “clk”. In downstream of the port “clk”, there possibly can exist a clock path through some combinational logic to a pin on which generated clock “gen_clk” has been defined. “gen_clk” is defined with a relationship such that the master clock acts as “Clock_1”. In we consider this simple scenario and perform abstraction on this circuit to extract a timing mode, then the generated timing model in the form of Liberty format will portray generated clock as shown in the right side of box of Figure 1. While we are able to model generated clock here well with information where the master pin is and where the target pin of generated clock is, what we have lost in this modeling is the generated to master clock relationship. Downfall of this makes the timing model unreliable for timing analysis. Such models when used for analysis of full chip timing analysis can associated the master clock relationship with any of the clocks defined at the master pin. Since in this case, two clock signals would be arriving at the master pin, in the absence of this missing relationship, it is possible that model will end up using “Clock_2” as the master clock. Besides rendering the whole analysis wrong, this leads to situation where one may have to consider design changes to extract a timing model which is reliable. Considering the complexity of today’s designs where hundreds of clocks may be at play in a given design, this can be huge challenge for the designers to extract a timing model.

To further complication the situation, we can consider a pipeline of generated clocks which are asserted in the downstream of one after another in a cascaded fashion and at each pin, there are possibly multiple generated clocks asserted. Abstraction of such timing models not only is a challenge for the designer but also for the software to ensure that the relationships are maintained as much as possible.

Second important aspect that we want to cover as a limitation of abstraction is the usage of path exceptions [4] in the timing model generation and associated pitfalls. While abstraction helps preserve the intellectual property since my logic is completely hidden, there are also pitfalls of using it. One




```
create_clock -name CLK1 -period 6 -waveform {0 3}
create_clock -name CLK2 -period 10 -waveform {0 5}
set_input_delay -add_delay 1 -clock CLK1 [get_ports {IN}]
set_output_delay -add_delay 12 -clock CLK1 [get_ports {OUT}]
set_output_delay -add_delay 1 -clock CLK2 [get_ports {OUT}]

set_multicycle_path 2 -from Flop1/Q –clock_to CLK1
```

Fig. 2

To illustrate above point, let us consider the circuit shown above. In the above circuit, we have an output port “OUT” and it has two output delay applied with respect to input clocks defined as “CLK1” and “CLK2” as defined in constraints in the right side box of above figure. Here, let us assume that the worst arrival to port OUT is via Flop1 (path1). If we apply a multi cycle path on CLK1 via Flop1/Q, the path from “CLK1” to “OUT” is now going to consume two cycles and data latching will also happen accordingly. A possible side effect of this would be on Path2 which can now become more constraining since Path1 has been relaxed with 1 additional cycle that has been provided to it via multi-cycle path constraint. Extraction of above circuit will show us a path from “CLK1” to “OUT” and also from “CLK2” to “OUT”. Due to abstraction, since we would remove all the internal pins, the multicycle path exception either now will show up for both the paths or will show up for none. In either scenario, we will have a discrepancy that one of the path fails to meet the timing slack when timing model is used in chip level context.

Accuracy of timing models is of paramount importance. Timing model since abstract out the information in the form of a library arc, there are multiple places where the accuracy loss is seen. We focus mainly on the characterization points for model generation. Characterization points need to be chosen for ensuring that the generating timing model is good for a range of slew and loading conditions. Some loss of accuracy is always there depending on the points chosen and it is mostly a trade-off between the number of points i.e. bulkiness of model vs. accuracy.

CONCLUSION

There must be a conclusion section that explains the paper's benefits, drawbacks, and potential uses. Don't use the abstract as the conclusion, even if it could summarise the paper's key ideas. A conclusion might go into more detail on the significance of the work or provide extensions and applications.

REFERENCES

[1] Hitchcock, R.B., Sr., Smith, G.L., Cheng, D.D., Computer hardware timing study, IBM J.
26 (1982) 100-105 Res. Develop.
[2] "Timing Abstraction of Intellectual Property Blocks" by S.V. Venkatesh et al., CICC'97, May 1997.
[3] Rakesh Chadha and Jayaram Bhasker: "Timing analysis for Nanometre Designs: A practical approach" (Springer, 2007) 190–200
[4]Vlado Vorisek, Bruce Swanson, Kun-Han Tsai, Dhiraj Goswami: Enhanced Management of False and Multicycle Paths in ATPG. VTS 2006: 160-165