Global approach to timing margins and constraint definition for SoC design, Part 1

Pierpaolo De Laurentiis, ST Microelectronics - December 22, 2015

Introduction - Design margins and new challenges

With the latest deep sub-micron technologies, System on Chip (SoC) design is becoming more and more troublesome due to

- Increasing size of design
- Increasing complexity of system
- Increasing complexity of work environment

Each of aforementioned items introduces challenges, especially for what regards the definition and management of design margins, which are the extra-pessimism added in the implementation cycle in order to guarantee that the device, once out of fab, fully covers the design specification.

The dependence of yield in respect to design margin is depicted in Figure 1. Below a certain level, yield is destined to be 0, because physical phenomena that occur, and may only become evident after the design phase, in real life devices impair the majority of paths. Increasing margin is going to make the design more robust against those phenomena but also is going to increase the cost of design, in terms of area and power consumption of the chip, and for what concerns the design effort needed to complete the design. Above a certain level, the design is simply impossible.

- Size of design. A huge chip in deep sub-micron technologies means there are a huge number of instances and a massive number of timing paths to be closed. This in turn leads to define design margins above the falling knee of the yield curve (see Figure 1), in order to get an acceptable loss of yield, staying very close to the best achievable with the current technology and design complexity. Also, it is needed not to approach the zone where the design cost begins to increase rapidly, to avoid any hazard versus the performance in terms of design time, area and power consumption.

Mathematically speaking, the designer should minimize the function of Yield/ProductCost.
For these reasons, design margins must capture the right statistics for the totality of timing paths in the design. More precisely, they should not be stitched to the worst case path (resulting in too costly a design), nor to the more forgiving ones (resulting in a design not robust enough). They should capture the majority of paths, while excluding the most exceptional cases. It should be clear that the curves in the graph in Figure 1 (and hence the optimal design margin) depend on the design style and design options/choices. As a consequence, when facing a brand new technology, any possible improvement that directly impacts margins should be introduced in the flow as soon as possible, because, as will be seen, even the early definition of margins, if overestimated, will inject impairments in performance. And those impairments are very awkward to remove later or will cause waste of resources due to overdesign.

- **Complexity of system.** System complexity often increases in terms of number of hierarchical instances, level of hierarchies, number of clock groups, number of modes, and so on. This poses the challenge of defining timing margins at each crossing of hierarchical levels to maintain coherence at each stage of design for maximum performance, while avoiding the problem of getting stuck at some phase because some margin was underestimated at previous phases. Also, for the same reason, a proper estimate of effects coming from a higher level or a different level of hierarchy must be inserted from the very beginning of the design.

- **Work environment.** In complex chips design, the required high level of specialization for each task tends to make the work flow contrived and fractionated, as each person involved confronts a tough task, full of technical obstacles, and may forget to stay in communication with other people on the team. To compensate and make the design process profitable and predictable, the entire design organization must standardize design margins and use leading-edge tools and methodologies. A central distribution of tools that calculate, propagate, and verify those design margins becomes necessary, too.

The frequent pitfall to be avoided is when a design team (which may be a third party) works separately on block development using different margins, resulting in a timing violation to show up when that macro is plugged in the top level for the final timing check. Because of this, the design of
the same macro would need to be opened for a number of fixes, which might be huge in costs of time and effort.

Any possible design customization that reduces uncertainty is welcome as long as it is embeddable inside the existing flow and tool-sets adopted by the design community. As an example, you can refer to the Low Uncertainty Clock Tree methodology for advanced clock tree architecture listed in the references below (LuCT) \(^6,7\).

Now we’ll discuss the definition, usage, and management of timing margins along all the design cycles. For sake of simplicity, we will not treat the issue of orthogonal problem of false paths.

**Addressing design margins for timing closure**

The approach to define, calculate, and propagate timing margins into design should always be inspired by the maximization of the Yield/ProductCost term. Consequently, improvements in the refinement of calculation or estimate model should be pursued throughout all phases of design. Also, any performance driven customization must be reflected in the timing margins’ management flow.

Another principle driving timing margins’ definition is to catch the phenomena for most of the samples with accurate statistics. Exceptional cases must be kept out, as long it is possible to recover them in the following part of the design flow, where the timing calculation is performed more and more precisely and on a single path base.

Over-conservatism induced by emotion must be avoided. SoC designers tend to add exaggerated hold margins, simply because in the past some design was harmed by hold failures, which proved impossible to recover on the device. This is an error because it forces the P&R tool to add a huge number of buffers everywhere just to close hold check, even where it is not needed. The consequence is that the entire design gets burdened by an increase of area, power consumption, routing congestion, high risk of electro-migration, and other issues.

Another frequent error happens when designers add an unreasonable amount of PLL jitter, simply because in the past a certain design had to be limited in performance after tape out.
Figure 2: General SoC timing closure flow

Figure 2 shows general SoC implementation flow, made of 4 stages, which concludes at SignOff (SO).

Physical design of a generic block of a SoC begins with the Synthesis stage, where impairments due to clock tree (skew) or other placed cells (crosstalk, delay of interconnect) are unknown. Hence, in synthesis the designer has to add a certain amount of uncertainty to cover the above parameters and to be sure the design process can proceed with no blocking points. In PreCTS (or placement) stage, the same designer can determine the effect of timing degradation due to placement, but not the effect due to the clock tree, which is going to be implemented in the next step, PostCTS. Hence, the effect of placement must be removed. In the following step, all the effects related to the clock network are going to be removed and so on, until SignOff.

The outcome is that all the items composing the timing margins must be made dependent on the particular design phase, and the process continues in the direction of removing something at each stage.

Main components of the uncertainties, which have to be entered in setup or hold timing checks, are
- PLL jitter
  - Variation of Clock Period
- Crosstalk (jitter)
  - Aggressing signals inducing timing variation on Clock Edges
- Dynamic IR drop (jitter)
  - Noise in power rails inducing timing variation on Clock Edges
- Local Skew
  - Margin related to clock skew
- On Chip Variation (OCV)
  - Variation of delay properties on clock network
- Clock Gating
  - Effect of gating on different branches/edges
- NBTI (data path)
  - Margin to estimate aging for specified mission profile

More specifically, physical details about real implementation aspects (such as insertion delay, skew, and crosstalk) are not known at the initial logical syntheses. As a consequence, estimates of these aspects are needed based on design statistics (area, size, number of registers, et cetera), known to designers at the beginning of the implementation process.

As the design process advances, different timing margin contributors can be cancelled, since the tools performing Timing Analysis (TA) calculate them. The design phases, from a timing margin’s perspective, are shown in Figure 3. Consider, as a couple of examples, the case of crosstalk that is removed from the list in the SignOff phase, where it is calculated by the TA tool, or the case of OCV on clock, which needs to be added in timing margins as the clock tree is not laid out, and then removed during as well as following the PostCTS phase.

It’s very important to consider, from the very beginning of the design process, all possible effects on the clock, which come from the top level. All these effects originate from the propagation of the clock signal outside the block, which has to be committed with timing constraints. Missing their definition might cause violations to bulge in the final timing analysis check, which is a calamity from a design process point of view.
PLL Jitter

The variation of “Tn” in the above diagram (Figure 4) is known as single period jitter, which gives the minimum time period.

Jitter can be specified in the form $\pm (A + N \times B)$, where

- $A$ is a coefficient representing bounded jitter due to disturbances such as supply noise,
- $B$ is standard deviation (1 sigma) in the normal distribution of the PLL output clock cycle time and it represents random variations,
- $N$ is the sigma multiplier and has to be set depending on the application requirements, i.e. admitted error rate.

In the example of chip design for the networking market, where zero failures in time (FIT) must be guaranteed over 10y operations, as shown in Figure 5, $N$ should be set at 9. This number can be relaxed under three conditions:

- if the frequency is much less than 1GHz,
- if the switching activity of a certain timing path, i.e. input data of the capture flop, is much lower than 1%,
if the mission profile of that path is such to wake up that datum for a very short time compared to the device life.

All those concepts can be condensed referring to the average switching frequency of the input data. As a consequence, a precise, and less pessimistic, tuning of the needed sigma multiplier might be given depending on the single path. This tuning is very awkward to accomplish during the first phases of a design, but it might give some chance of cleaning small violations during the final SignOff (SO) analysis.
Due to fluctuation of physical parameters, both launch and capture clock edges timings in ‘register to register’ (reg2reg) paths internal to the block are subject to some uncertainty. These variations are proportional to the clock Insertion Delay (ID). Shared parts of the clock tree do not contribute to reg2reg variations, as they cancel out, according to the Clock Reconvergence Pessimism Removal mechanism (CRPR). Before clock tree synthesis, the uncommon part of the clock distribution is conventionally forecast at a percentage of the relevant block ID. The OCV value to be chosen in synthesis must be the same used in final SO, if a fixed derating coefficient is applied. If, instead, the SO task exploits the most recent techniques such as AdvancedOCV (AOCV) or ParametricOCV (POCV), where the derating factor is made dependent on the path depth, or the statistical spread coming from the local mismatch is calculated on the single instance base, an opportune choice must be made, according to the principle of Figure 1. As indicated in Figure 6, it is opportune to define a fixed derating for the synthesis and PreCTS cases, which is not stitched to the first level defined in AOCV table (the most pessimistic one), nor to the plateau value.

This fixed OCV might correspond to a value defined for a low-medium depth in order to lose a
negligible amount of conservatism in respect to short paths and not be exaggerated over long paths, the latter being the most troublesome.

Figure 6: Defining a fixed derating for the synthesis and PreCTS cases

In the SO phase, exploiting the aforementioned advanced techniques for timing analysis (AOCV, POCV) reduces the pessimism by analyzing the single path at the maximum amount of accuracy, making it possible to easily analyze and possibly fix a small number of violations.

As shown in Figure 7, in the example of POCV, the local mismatch can be computed as function of input transition and output load on a single instance base, once the proper characterization libraries are available.

Figure 7: Statistical spread in timing as function of transition and load

NBTI

NBTI

- Derating may also be needed in data paths to compensate for differences in delay between existing aged libraries and the required lifetime of the device.
- This implies a further derating in propagation delay.
Only setup check is affected—margin is given as a percentage of clock period (Tck).
The margin is not needed if libraries covering the targeted mission profile are available.

**Latency estimate**

- Based on real design data collection and statistical analyses, latencies in blocks can be directly linked to the number of registers.
- For cases where the design is congested or awkwardly shaped, a multiplying factor is given, depending on shape.
- Effect of clock gating can be quantified similarly.
- All OCV, crosstalk, dynamic IR-drop estimates depend on the latency value. Of practical importance is the ability to refine and receive feedback as soon as possible for more precise estimates or real value from implementation, also for what regards top level ID, in order to better calibrate all quantities depending on estimates.

![Figure 8: Latencies in blocks can be directly linked to the number of registers](image)

**Crosstalk, dynamic IR drop**

- Crosstalk by adjacent signal lines and noise on power rails can alter the quality of the clock.
• Depending on the used TA tool, crosstalk can be computed in SO.
• If estimated, crosstalk effect is a percentage of top insertion delay, depending on the type of clock tree. If LuCT is adopted, as an example, the crosstalk is very nearly 0.
• Dynamic IR-drop effect is a function of top ID.
• Hold checks need to compute just the uncommon portion of paths.
• For hold, it is also possible in SO to refine constraints on a single path base to make a final cleaning of small violations.

**Mixed Aging**

• Depending on the min/max delay type, aged or fresh libraries can be selected. In particular for setup check, the aged library is used for the launch node time evaluation and the standard non-aged library (0 years) is used for capture node time evaluation.
• The effect of mixed aging has to be estimated only in case of a gating that differentiates the aging of two branches:
  - Gate only in one branch
  - Opposite edge case and gating

*Part 2* will discuss how to address complexity.