Managing Derivative SoC Design Projects to Better Results

Lane Albanese
ReShape Inc.
lane@reshape.com

Abstract

Many companies have moved to implementing a “platform chip” or derivative design strategy to preserve their investment in design tools. But physical design is proving to be a broken link in the efficiency chain. In most cases, SoC derivatives are completely new from a physical design standpoint. This paper presents techniques and tools a design manager can use to improve predictability, quality and throughput, including a Design Maturity workflow model that breaks down the SoC design process into four successive standard phases.

1. Introduction

“80% of design is redesign” and it is especially true today in the system-on-chip (SoC) age. Not only is it expensive to design these complex monster chips outright, but there is associated software, programmable and embedded, in and around the chip. Unwilling to discard the hardware and software already invested upon and start everything again from scratch, many companies have moved to implementing a “platform chip” strategy. From a platform chip, modifications to the chip design are made to target different market segments. This approach minimizes the development cost associated with architecture design, software development, and standard component block creation. It is no surprise to conclude that efficient derivative SoC design is a necessity to successfully compete in today’s market.

The broken link in this efficiency chain is physical design. The problem begins with understanding that the term “derivative” can mean many different things, and it can mean vastly different levels of rework are required depending on which phase of the design process is affected. Adding an additional DAC or adding a new piece of soft IP to an existing design may be architecturally trivial, but can result in a huge amount of physical design work. In most cases, SoC derivatives are completely new from a physical design standpoint. Further decreasing the efficiency is the fact that physical designers have often found it difficult to reuse their work, causing the layout of the platform chip to seem like a bottom-up, start-from-scratch, project. Additionally, it is a major design management challenge to juggle the growing problem of design complexity, as well as the intricate details associated with nasty nanometer design issues such as signal integrity, IR, and electrical migration.

2. Challenges Facing Physical Design Managers

Physical design managers responsible for SoC design confront a growing mountain of challenges. Increased design complexity requires more people and moving to hierarchical physical design, which means more licenses, computers, and data to manage. The issues are compounded by the tyranny of details: nanometer effects call for additional verification steps and the associated new tools have to be purchased and deployed.

When life was simpler (though it didn’t seem that way at the time), you could manage in an ad hoc way; you didn’t worry about porting anything to the next design, and you could afford to throw people at every new problem. That won’t cut it in the SoC platform chip world.

Success dictates that you become competent at solving four key issues:

- Developing realistic project plans for the large SoC projects
- Putting together the right resources (team skill composition, machines and licenses) and managing them efficiently
- Measuring progress in a way that leads to “on time, on cost, on spec”
- Reusing the methods and knowledge that were gained from prior designs
2.1 Planning

The huge number of tasks necessary to complete a large physical design project must be identified and described, yet the amount of details to manage creates a significant potential for error. It is the unpredictable nature of the physical design process that makes planning so difficult. The old army adage is “all preparation plans go out the window at the first encounter with the enemy.” And this is equally true for chip project planning.

One way to deal with (a) the need to plan and (b) the difficulty with planning is to create both a global project plan and detailed plans. The global project plan will identify the major milestones and dependencies. Based on the experience with prior versions of the SoC, the global plan can also provide a schedule estimate for a tape out date. Trying to predict a specific date for tape out is something that may appease upper management, but is virtually impossible to do at the beginning of a large SoC project. A better approach is to specify a probable target date with a reasonable range of time preceding and following the target date where tape out is likely to occur. Detailed plans should be formed for smaller time increments, e.g., two-week intervals, and you should expect them to be dynamic.

2.2 Resources

The complexity of a large physical design project makes it difficult to accurately and confidently plan resource requirements. For instance, a physical design team manager may find that he does not have sufficient expert engineers to complete his project within a reasonable amount of time. The manager may find that his engineers’ areas of expertise are not the ones needed. Additionally, the manager may find that use of people, machines, and EDA tool licenses is inefficient because he cannot predict when these resources will be needed or how they should be deployed.

2.3 Project measurement, assessment, and adjustment

Once the project commences, the most critical task for the design manager is to understand the state of the project with respect to the plan. This requires the ability to measure what has been accomplished, assess the progress against the plan, and finally make adjustments as necessary. The measurement and assessment can be daunting on a complex SoC project. One needs to know what has been completed, and more importantly, the quality of what has been completed. There needs to be a way to quickly and repeatedly “measure” the state of the project.

2.4 Project Reuse

At a high level, little design environment infrastructure is used from design to design. This is especially true if derivative designs are done by various groups in different geographical locations. At a lower level, specific scripts tend to be customized for a specific design and may not be too useful except to the person who originally wrote them. Much has been written about design reuse in software and RTL design, but it has been largely non-existent in physical design. Being the last group in the chip design chain, the physical design team is often in the pressure cooker state of having to compensate for any other earlier team’s schedule slips and just finish the chip. It is a rare physical design team that can afford the 50% additional resources to make environments and scripts reusable by engineers doing the next derivative.

3. Solutions

Techniques and tools the design manager can use to improve predictability, quality and throughput are:

- A workflow that is cognizant of the maturing nature of a design over time. RTL and physical design are designed in parallel. Milestones are defined by the maturity of the netlist, IP, and layout.
- The right team. People are everything on a project. The right people utilized in the correct way will yield the greatest chance of success.
- Fast iterative design to accurately assess progress. This technique is similar to large software projects that compile and run regressions every night. One frequently builds the full-chip design and measures the results. One can exactly measure the quality of the design at any point in the project with little cost. Tools are emerging to automate the GDSII construction process that are much more automated than traditional scripted environments.
- Project reuse. Tools are becoming available to make reuse practical in physical design.
4. Design Maturity Workflow Model

The underlying assumption in the Design Maturity model is that the RTL, IP and chip physical design are all being designed concurrently. Each team is refining and correcting their respective part of the design. In a traditional concurrent model as shown in Figure 1, the physical design process doesn’t begin in earnest until after the other parts of the design process has begun. The physical design team receives the front-end design team’s partially completed work. They know they have to live with changing design attributes (netlist, IP, specs) until the tape is shipped off to the mask shop. The physical design team’s work will often continue for a significant amount of time after the “front end” work is complete.

![Figure 1. Conventional concurrent design.](Image)

The front-end and back-end teams design in parallel. Two or three times in the design there is a full-chip build milestone where the back-end team can assess the GDSII quality and determine the project status.

To improve the traditional concurrent workflow, we have defined a four-phase design maturity model. This model is at the heart of the project planning and measurement process. Each phase begins with the deliverables appropriate to that phase. Team activities specific to the respective phase carry the physical design process towards its goals, which are the deliverables for the next phase of the physical design process. A phase is concluded when all goals for that phase have been achieved. It is quite common for overlap in phases to occur for varying components and goals of the design.

![Figure 2. Design maturity model.](Image)

The SoC design project is divided into four phases. The milestone that defines one phase from another is the characterization of the maturity of the design's netlist, IP, and physical design attributes. Using this model, management can accurately assess where they are in the project and whether schedule dates are realistic.

The Design Maturity workflow model breaks down the SoC design process into four successive standard phases: Setup, Convergence, Refinement, and Closing.

The following chart (see Table 1) describes the start of phase deliverables and goals for each phase. The deliverables, goals, and phases themselves may be adjusted to suit the requirements of a specific derivative or “from scratch” SoC project. The deliverables and goals (milestones) along with the associated dependencies may be entered into a standard Gantt chart to a level of detail desired by the design manager or project leader.
Table 1. Deliverables and goals for each phase of the Design Maturity workflow

<table>
<thead>
<tr>
<th>Phase</th>
<th>Setup</th>
<th>Convergence</th>
<th>Refinement</th>
<th>Closing</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Set up the rest of the project. Coordinate the timing of specific events and assemble resources required to complete the project.</td>
<td>Stabilize design enough to start building the chip. Build the full-chip from deliverables. Multiple netlist and IP iterations are expected.</td>
<td>Realize the final SoC specification.</td>
<td>Close out the design; complete the project to tapeout to the mask shop.</td>
</tr>
<tr>
<td>Start-of-Phase Deliverables</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Netlist</td>
<td>Initial</td>
<td>Structurally Complete</td>
<td>Almost final. Final very minor design tweaks are still possible.</td>
<td>Final</td>
</tr>
<tr>
<td>IP, includes cell libraries and memory elements</td>
<td>Initial</td>
<td>Updated</td>
<td>Final versions</td>
<td></td>
</tr>
<tr>
<td>Specifications</td>
<td>SoC Info</td>
<td>Updated (I/O, clocks, timing)</td>
<td>Final I/O</td>
<td></td>
</tr>
<tr>
<td>Constraints</td>
<td>Project (e.g., die size, power goals)</td>
<td></td>
<td>Final timing</td>
<td></td>
</tr>
<tr>
<td>End-of-Phase Goals</td>
<td>• Project plan to lay out the remaining physical design process</td>
<td>• Top-level stabilization (of the whole SoC design).</td>
<td>• All blocks are routable.</td>
<td>• Final full GDSII build with all design for manufacturing features implemented (antenna check, metal fill, etc.)</td>
</tr>
<tr>
<td></td>
<td>• Project setup, including directory setup and software installation</td>
<td>• Pads</td>
<td>• Timing is closed or closable for all blocks (including the top-level block).</td>
<td>• Minor timing ECOs</td>
</tr>
<tr>
<td></td>
<td>• IP setup, checked to determine that IP is consistent and correct</td>
<td>• Power Grid and Structure</td>
<td>• Most blocks are DRC clean or cleanable.</td>
<td>• Final analysis views of chip created</td>
</tr>
<tr>
<td></td>
<td>• Netlist analysis, including the examination of design, tool use and possible issues.</td>
<td>• Floor plan</td>
<td>• Most blocks are LVS clean.</td>
<td>• Final tape out database created</td>
</tr>
<tr>
<td></td>
<td>• Trial floor plans, which forms the initial plan of the SoC's design.</td>
<td>• All or most sub-blocks are place and routable.</td>
<td>• All EM/IR issues are resolved.</td>
<td>• Final verification runs and fixes.</td>
</tr>
<tr>
<td></td>
<td>• Trial block level builds in which work is begun to build individual blocks on the SoC</td>
<td>• Initial timing runs and constraint refinement.</td>
<td>• Top-level is frozen.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Initial power rail (IR drop) analysis.</td>
<td>• Top-level stream out.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Formal verification passing</td>
<td>• Initial Top level LVS/DRC/ANT runs.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Design Rule Check (DRC) and Layout</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Versus Schematic (LVS) attempted at the block level</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Die size convergence.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
5.0 People and Team Structure

The team structure for conventional hierarchical physical design has individual blocks assigned to certain engineers, who then are responsible for their blocks for the entire workflow.

Using a fast iterative design methodology for moderate SoC physical design, we have found it effective to instead organize a team as follows. The core team should be comprised of:

- A top-level engineer, who has the final responsibility for top-level engineering, including pad, power, and floorplan.
- A timing analysis engineer who is responsible for overall chip timing closure.
- An analysis engineer who has a broad area of responsibility with physical verification, timing analysis, and power analysis.

The physical design team may be larger depending on the following project requirements:

- Die size and number of place and route blocks
- Clocking strategy and the number of clocks
- Custom work and the handmade aspects of the SoC design process
- Quality (analysis required)
- Project goals
- Schedule

An important point is that the core team in place at the beginning of the project should be small. These members do the up-front work and exploration needed in the setup phase, when the design is usually fairly immature. Additional people with the needed expertise can then be added as needed. The peak team size will most likely occur during the convergence phase when blocks are frequently built, analyzed, tweaked, rebuilt, etc. Ideally, the team size should begin to taper down in the refinement phase, when certain people are hammering out the last few problems.

The individuals on the team might play the following roles with particular tasks and responsibilities:

- A chip lead provides the required strong technical leadership and has the overall responsibility for the chip design. This individual might not have enough time leftover to work on the chip itself.
- The top-level engineer has the final responsibility for top-level engineering, including pad, power, and top floorplan.
- Two place and route engineers are responsible for place and route on the chip.
- The timing analysis engineer is responsible for timing constraints and simple verification.
- The analysis engineer has a broad area of responsibility with formal verification, timing, EM, and IR analysis.

6.0 Fast design iteration is the enabler

In conventional physical design workflows, physical design teams are handicapped by having to constantly return to the front end of the project in order for the rest of the project to unfold. In other words, whenever the development team encounters a problem with the chip design, they have to stop working and go back to the front-end of the project to correct the original condition that gave rise to the problem. This continual rework process is the major cause for delays in physical design. And it’s why most teams are reluctant to work in concurrent mode – change means, in effect, fill in the hole you just dug.

The simple fact is that you do not really know where you are on your chip project until you have built it using your sign off tools, and just as important, have analyzed those build results. It is only then when you can run your analysis tools to really know where you are. In a conventional flow, this step is labor intensive; it can take three, six or even nine weeks to assemble the full chip for a complex SoC.

The solution to this problem is to automate chip construction so that you can build the chip overnight. This is exactly what RTL and software teams do: recompile their code overnight. A similar 24-hour construction capability has only been recently available to physical designers. By building and analyzing the results more frequently, a tighter iteration loop can occur with the front-end team realizing a significant schedule savings.
Figure 3. Fast 24-hour builds allow for earlier feedback from the back-end to the front-end design team.

7. Project Reuse

A primary shortcoming in physical design is that there is little physical design knowledge reuse from project to project. Without some team continuity, reuse pretty much starts and ends with “hard” macros. In addition to having at least one individual from the old team on the new team, derivative chip design needs reuse of the following:

- Tool knowledge. There needs to be a way to share how to use the tools and tool tricks, i.e., there needs to be “designer reuse.” Ideally you’d like to abstract away the need to know how to drive tools so that you are not dependent on one tool expert that can quickly get assigned to too many projects and quit the job.
- Construction recipes. The recipe for building a block could remain the same even if there is a modified floorplan for the new design, e.g., layout aspect ratio is reshaped.
- Floorplans. Even if the aspect ratio or dimensions of blocks are changed, the placement recipe could stay the same. This would eliminate the burdensome labor of re-floorplanning in absolute coordinates.
- Standard design environment. An environment with standard directory structure and naming conventions enables team members to navigate the physical design process and find relevant data with ease. Standardized design of SoCs and blocks provides instant familiarity with chip and block design for physical design teams.

By addressing these four needs it would become possible for physical design knowledge from prior projects to be easily transferred to and modified for new projects.

8. Case Study: 10M gates, 225MHz, in 20 weeks

This SoC project was primarily intended to be a productized version of the initial design that was fabricated. While there was some new and updated hardware IP incorporated into the design, the focus of the efforts was to insure that all design quality criteria were met in this second version of the chip. Specifically, timing closure including crosstalk effects and robust EM/IR analysis were objectives that had to be achieved. As is usually the case, schedule was also of paramount concern.

8.1 Planning

Although the planning for this project was based on the design maturity milestones, it was also unique in that this version of the SoC was a limited redesign of the prior version. Many input design deliverables were able to reach a higher level of maturity earlier than in a traditional concurrent design, but there were deliverables such as final timing constraints that followed the typical pattern of not being finalized until the design was almost ready to tape out.

The original schedule was broken down as follows:

<table>
<thead>
<tr>
<th>Phase</th>
<th>Duration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Setup Phase</td>
<td>4 weeks</td>
</tr>
<tr>
<td>Convergence Phase</td>
<td>3 weeks</td>
</tr>
<tr>
<td>Refinement Phase</td>
<td>6 weeks</td>
</tr>
<tr>
<td>Closure Phase</td>
<td>3 weeks</td>
</tr>
<tr>
<td>Total</td>
<td>16 weeks</td>
</tr>
</tbody>
</table>

In a more traditional concurrent design, the setup phase would be shorter and the convergence phase would be longer than shown above. The schedule above was based on the fact that the physical design process was started mid December (generally a poor time to start a project) and hence the project would not hit full stride until early January. It also considered the high quality of
the initial deliverables, which were the final outputs from the prior version of the design.

8.2 Resources

The project team was structured as follows:

- **Project Lead Engineer**
  - Day to day technical coordination of the project
  - Overall responsibility for timing closure
- **Top-level Engineer**
  - Padrng, power grid design, pre-routes, overall chip floorplan, full chip DRC/LVS
- **IP Librarian**
  - IP inspection, library creation, DRC/LVS of IP
- **Place and Route Engineers**
  - One for high speed interface block
  - One for clock distribution module and top-level clock distribution
  - Two for other place and route blocks
- **Timing Engineers**
  - One for high speed interface block
  - One for overall timing closure
- **Analysis Engineers**
  - Two for tasks such as formal verification, EM/IR, DRC/LVS

The team utilized the ReShape hierarchical floorplanning, optimization, and automated build environment. All project source data was under configuration management using Perforce. The underlying place and route tool was the Astro suite from Synopsys. PrimeTime SI was used for timing analysis, and Mentor Graphics Calibre was utilized for physical verification. The server pool was primarily high performance Linux servers, although Sun servers were utilized for those jobs requiring large amount of memory. LSF was the queuing environment.

8.3 Project temperature readings

Each Design Maturity model phase of the project required certain criteria to be achieved before continuing into the next phase. In some cases the criteria applied to the entire design, while in other cases it was applicable to individual place and route blocks. To accurately determine whether a milestone had been met, we built full-chip and/or block-level GDSII using the production place and route tools and performed the appropriate analyses.

On this project, ReShape’s PD Builder facilitated rapid full-chip GDSII builds from revised netlists and/or modified design specifications. The fast turnaround time was achieved by partitioning the design into hierarchical blocks no larger than approximately 250K placeable instances. All blocks were placed and routed in parallel on fast Linux machines. PD Builder also allowed a variety of analysis tasks to be automatically run after the block builds were complete. The results were data mined and prepared into concise reports that gave a clear understanding of the state of the chip for that specific build iteration.

By using PD Builder the team was able to rebuild the design from netlist two to three times per week to run analysis tools such as PrimeTime SI, Formality, AstroRail, and Calibre. This rapid and accurate feedback enabled the team to uncover issues as early as the setup phase.

8.4 Project Reuse

The team used a standardized design management paradigm. As previously noted, all design data was source controlled using Perforce. PD Builder provides a standardized directory structure and naming conventions. The standard build environment allowed anyone of the project to rebuild any part of or all of the design. PD Builder also supports a common physical design environment that was used from reading in the initial netlist through implementing final stage timing ECOs. Because of the above factors, a follow-on project to this SoC that also used PD Builder was able to highly leverage and reuse the knowledge that was gained during this project.

8.5 Results

The design taped out in 20 weeks, four weeks later than the original plan, but meeting all of the stringent quality requirements. This delay was due to a variety of typical factors including some late deliverables, the usual EDA tool issues, very late stage ECOs, and certainly some execution problems on the part of the physical design team. Overall, the project was viewed as a success and produced the high quality silicon that the customer required. The chip came back from the fab and worked across the temperature and voltage range.

- The design was completed in 20 weeks. There were unanticipated ECOs throughout the design; all were accommodated with minimal schedule impact.
- 165 man weeks of effort were required, about 30% - 40% less manpower than with a conventionally scripted design methodology with few iterations.
- PD Builder's abutted block support helped us achieve a 5% smaller die size.
- The chip was fully verified to be IR and EM correct.
- Timing was met at 225MHz.
9. Conclusion

The time has come to move physical design to the same level of productivity available to software and logic designers. The Design Maturity model provides an excellent way to plan and systematically measure where the design is. Matching the correct team, both size and skill set, to the SoC plan is also required to assure success and maximize the overall project efficiency. A fast iterative design style makes measurement easy and enables engineers to do what they do best – solve problems, while allowing design managers to make the necessary plan adjustments that always occur on a complex SoC project.