AMBA Interconnect Design Flow Automation

Tom Ajamian

Analog Devices
3 Technology Way
Norwood, MA
USA

www.analog.com

ABSTRACT

A system architect faces many challenges when designing an AMBA-based interconnect for an SoC. Gathering requirements, creating specifications, and then taking those specifications and turning them into a cohesive design that matches requirements for performance, power and area (PPA) is not straightforward. Quantifying PPA is essential in tuning the design for the requirements. While traditional pencil and paper or spreadsheet methods have been used for PPA estimates, these approximations fall short for tuning and not overshooting the performance of the design. Tools such as Synopsys' Platform Architect and its Interconnect and Memory Subsystem Optimization (IMPO) flow help provide a solution to quantifying performance; however closing the loop between specification and design using ARM's AMBA Designer and Platform Architect modelling can require manual translation. We propose a more automated flow, from specification creation to design to performance analysis that allows for the specification to be the sole data source for all stages saving considerable time and reducing the potential for error introduction.

Table of Contents

Table of Figures

Table of Tables

1. Introduction

A system architect faces many challenges when designing an AMBA-based interconnect for a System-on-a-chip (SoC). Gathering requirements, creating specifications, and then taking those specifications and turning them into a cohesive design that matches requirements for performance, power and area (PPA) is not straightforward. Tuning PPA is the dance between SoC/System architects and their design tools that happens over and over during the course of a SoC's design, and throughout this flow, specifications need to be captured and communicated to design and development teams. Often remote teams will drive specification changes which need to be wrapped back into the design. When these design changes are made, any SoC architectural modelling environments used must be updated to reflect the latest design. Keeping all these sources of truth cohesive can be a tedious, full-time job.

When working with an ARM AMBA-based interconnect such as NIC-301 or NIC-400, the interconnect configuration is captured in an XML file managed through AMBA Designer's graphical interface, from which RTL (Register-Transfer Level) is stitched [1]. A specification representing a higher level of abstraction can be used to describe external interfaces of the interconnect and their names, as well as data widths and clock domains. The SoC memory maps are usually described in spread sheet form, and can be hundreds of entries long. The translation between these high level specifications and AMBA Designer's graphical interface is entirely manual. Settings critical to interconnect performance typically are not captured in any specification outside of AMBA Designer's graphical interface of tabs and menus, resulting in a constant effort to maintain consistency between specs and the AMBA Designer XML representation.

Adding to this already complicated flow, SoC architectural modelling and analysis for any of the PPA variables requires additional steps. In addition to creating a design in AMBA Designer, an architectural model of the design is also required for analysis and optimization of performance. A simple spreadsheet-type analysis framework could be performed, using data widths and clock domains between masters and slaves to map use cases to the SoC's interconnect. However, these types of static analyses fall far short of finding actual bottlenecks in the interconnect, as dynamic interaction and queuing theory play a role in how the actual system functions. Without a quantitative way of modelling the interconnect and being able to feedback observations into the design, chances are high that the design will be not meet performance or be woefully oversized, increasing the power and area of the design unnecessarily.

Synopsys' Platform Architect MCO (Multi Core Optimization) provides a tooling environment together with performance models, methodologies and flows which allow early and efficient exploration of performance and power [2]. More specifically, one can import an AMBA Designer XML configuration and model the design using a combination of cycle-accurate SystemC/TLM2 models for the AMBA interconnect, generic cycle accurate and approximate models for masters and slaves, clock generators, and stimuli. Once the AMBA Designer XML configuration is imported in Platform Architect, this representative SoC platform must be populated with source and sink models (for masters and slaves), and stimuli must be provided for each of the masters represented by traffic generators. This platform population and completion must happen separately for each AMBA Designer XML candidate design. Finally, once

platforms exist for each candidate design, simulation and analysis of this architectural performance model is yet another critical and intensive task that must be performed.

Despite having modelling and analysis tools at our disposal, using tools for more than a handful of candidate designs can be time consuming. To address this challenge, Platform Architect MCO (PA) provides an Interconnect and Memory Performance Optimization (IMPO) flow which allows for efficiently sweeping simulation across the design configuration space and stimuli while collecting and collating analysis results from each simulation run (within a simulation sweep) in a spreadsheet form which then allows for efficient and powerful methods for performance analysis. However, coupled with the time spent on specification management across the design and the performance modelling functions, we see that there is a need for a more automated flow that starts with a single source – the specification which can drive both the design in AMBA Designer and also the creation of the architectural performance model for Platform Architect MCO.

2. The Challenge

To ease the designer's challenge in both working with an extended remote team and generating new designs, an ideal workflow should allow:

- Single-source specifications – a single source of truth
- Automating translation between specifications / design file formats, only during generation
- Detailed and accurate methods for performance analysis of the interconnect are required
- Modelling for performance analysis should be a by-product of the specification and design flow (e.g. automatically generated)

In the specific case for our team, we work closely with a remote design team located in Bangalore, India. With the majority of the designers in Bangalore, interconnect design and SoC architects in Massachusetts, and a software team in Edinburgh, Scotland, we felt the need for a clear communication system. As spreadsheets are ubiquitous, this seemed like the clear choice for a specification format for address maps and masters / slaves for the interconnect. For NIC-301, we combined the Synopsys' Platform Architect modelling environment with a framework we built for traffic stimuli and analysis, called IPA (interconnect performance analysis) [3]. With the above guidelines and existing systems in place, we were ready to implement an automation flow that met our needs.

3. Design Flow Methodology

Before delving into the full automation flow, it is important to understand the evolution of interconnect design and how the addition of particular tools and methodologies benefits this design process. The following sections explore details for flows ranging from basic to complex, with the final flow as the target of this paper's efforts.

Excel and RTL-based methods

Clock frequencies and data widths can be used as the inputs to a simple *static analysis* of a design. In this type of analysis, we take the data widths and clock frequencies of a master and slave combination and compute the theoretical maximum bandwidth. By building up a table of

such relationships as a reference, applications can be mapped to this reference and equations defined to quickly show if the application is asking more than theoretical from any interface. These analyses typically rely on a de-rating value, which de-rates the theoretical maximum throughput by some fixed percentage. For example, the architect may assume the system will only be 70% efficient – thus the theoretical maximum throughputs are reduced by 30%. While a useful tool, spreadsheet-based static analysis is not modelling the dynamic aspect of interconnect transactions and the peaks and valleys that come with these interactions. As such, it is more of a general data width and clock sizing tool, and not an interconnect tuning tool.

Using RTL for performance analysis, by nature, comes later in the design phase and is also gated by availability of stable RTL. Moreover, the instrumentation and monitoring within the RTL for purposes of architectural analysis may also be limited and/or gated by availability. RTL based simulation models are more suitable for performance validation before silicon tapeout or in some cases where complete accuracy is desired and a higher level SystemC/TLM2 abstraction model is unavailable. In such cases, one might explore co-simulating RTL with SystemC.

Complex SoC architectures require a dynamic simulation environment which will allow the SoC Architect, System Designer and/or the System Architect to efficiently model, simulate and analyze critical vectors of power, performance and area during "early" architecture and design phases of development. Synopsys' Platform Architect MCO provides such a desired environment which can allow performance exploration, optimization and validation of a candidate SoC architecture much early on in a product development schedule.

Out-of-box (OTB) AMBA Designer XML (AD) import and IMPO flow with Platform Architect

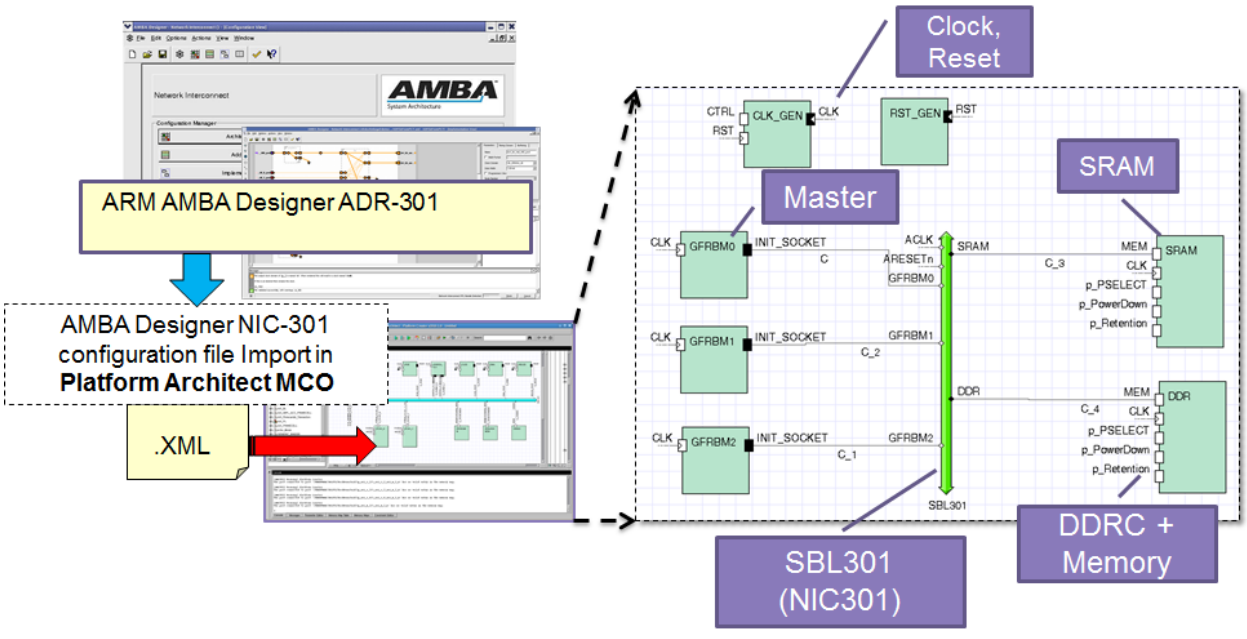


Figure 1 - AMBA Designer Import into Platform Architect MCO

Where existing tools and methods for modeling and performance analysis fall short, Synopsys' Platform Architect and its IMPO flow takes over. With the cycle accurate SystemC model, SBL-301 bus library model to represent NIC301 interconnect, and its built in analysis instrumentation and monitoring capability along with Tcl Command Line Interface (CLI), Platform Architect gives the system architect a higher caliber tool for the toolbox.

Shown in Figure 1 - AMBA Designer Import into Platform MCO above is a pre-configured AMBA Designer XML Import Flow for Platform Architect. In this flow an existing AMBA Designer XML configuration for an NIC301 interconnect design can be imported into Platform Architect to help quickly assemble an architecture performance model using the SBL-301 bus library model. Also shown in Figure 1 above is a screenshot of a simple example platform which contains generic models for traffic generator masters, slaves, clock and reset SystemC/TLM2 blocks. Platform Architect MCO provides automation for sweeping design parameters across multiple simulation runs. Once such a performance model has been assembled, the next step is to integrate the IMPO framework into the PA based simulation model for efficient sweeping of design parameters, exploration and performance analysis across the design space of interest.

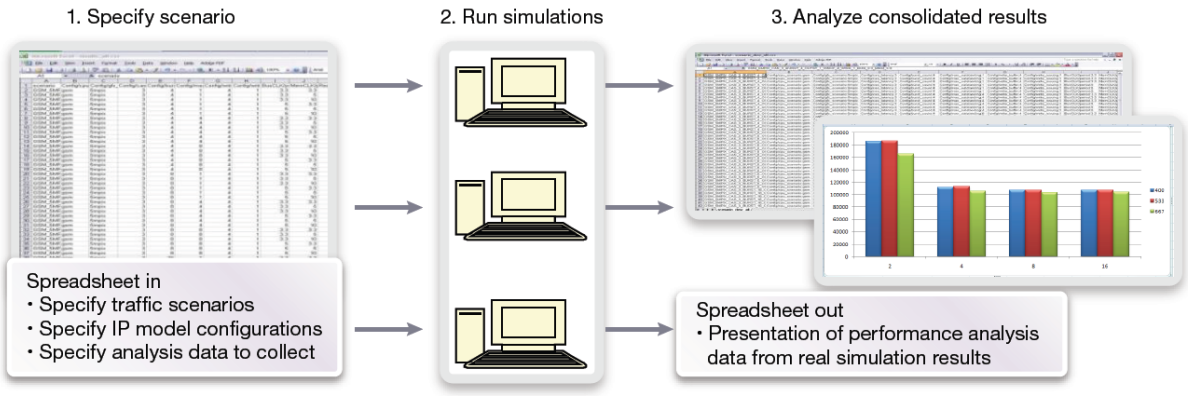


Figure 2 - IMPO Flow

As shown in Figure 2, the flow requires a spreadsheet defining the design parameter permutations for which simulations are run and for which analysis results are consolidated.

1. The input scenario specifies the set of simulations with their respective design parameter settings. A scenario is specified in terms of a Comma Separated Value (CSV) table. Each row represents a single simulation and each column represents a design parameter value.
2. Platform Architect MCO then executes the simulation as defined in the scenario file. The analysis results from each simulation are stored in a separate analysis database. Additionally, the top-level performance metrics are consolidated into a results table.
3. The results table is similar to the scenario file, but with additional columns for the analysis metrics. The results table can be immediately converted into a Pivot Chart using common spreadsheet tools like Microsoft Excel or Open Office.

Platform Architect MCO comes with a set of predefined metrics to support detailed and insightful bus analysis (latency, throughput, contention, and utilization) memory subsystem analysis. Arbitrary custom metrics can also be enabled via the Tcl scripting interface.

ADI's Interconnect Performance Analysis (IPA) Framework & PA IMPO

IPA is an internal framework built using various Python automation and data analysis packages and has been applied successfully to three ADI SoCs. While PA and IMPO are central to the IPA methodology, IPA is a substantial software framework to enhance and customize these capabilities in two broad areas:

- Application definition, exploration and modelling
- Post-processing and data analysis

The overall diagram of IPA's workflow is shown in Figure 3 below.

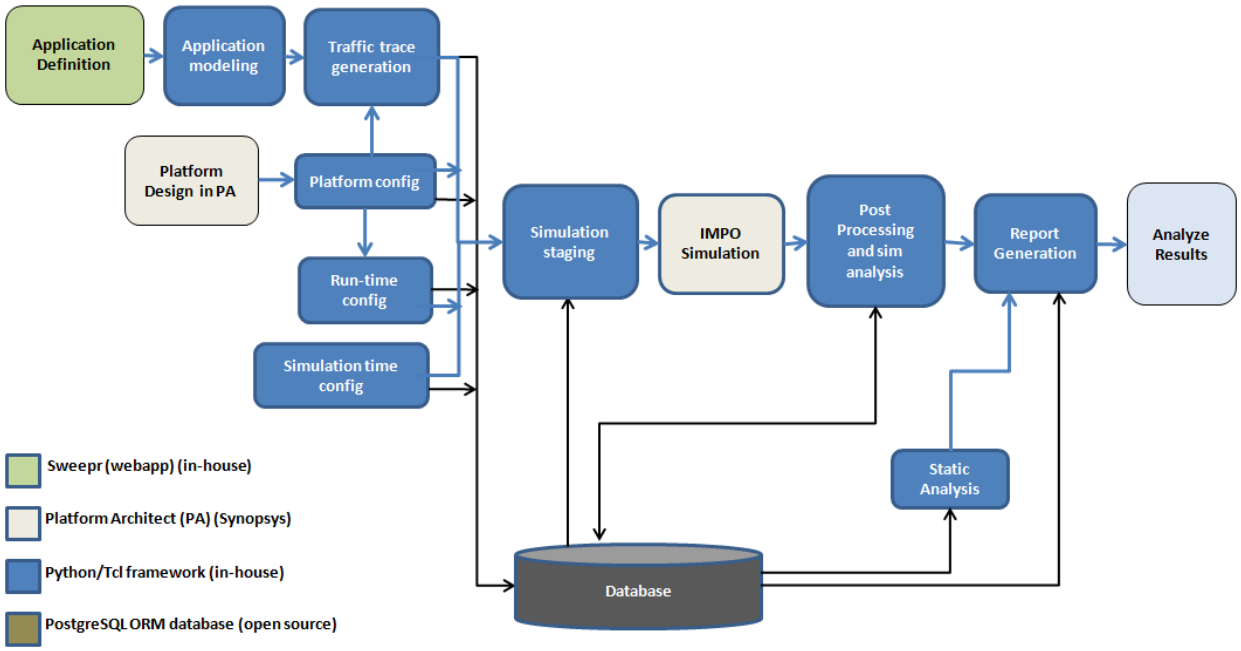
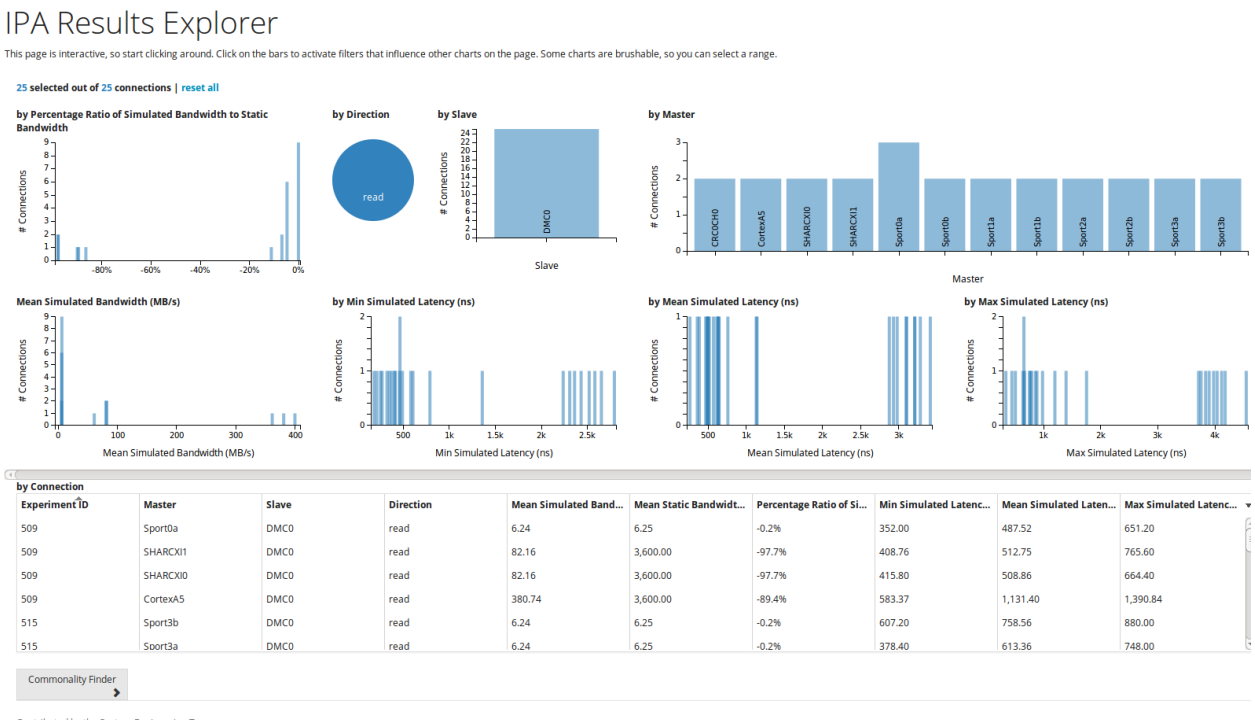


Figure 3 - IPA Automation Flow

IPA operates on single experiments, where an experiment is a combination of a unique set of a single PA platform specification, operating conditions, and PA variables, also called the hardware definition. Operating conditions include AMBA traffic parameters, abstracted into tasks. The hardware definition is a custom set of values that the user can choose to manipulate, sweep or permutate in the design. As an input, IPA can take a specific application driven top-down definition, which defines many experiments in a specific application search space. These are permutated against any user specified hardware configurations to create sweeps of experiment simulations, which can then be viewed automatically in a sensitivity analysis style report.

IPA has a baseline performance suite which enables fully automated one (master)-to-one (slave) and many-to-one experiments to be generated based on a platform specification. These are useful

during the initial analysis of an interconnect or when the use cases are quite broad or not yet defined. Experiments are generated based on a static analysis of the clock frequencies and data widths defined in the platform specification, configured to run up to their theoretical maximum bandwidth. To help ease analysis efforts, the web application IRE (IPA Results Explorer) was developed which allows the user to actively explore a one-to-one or many-to-one results set and quickly determine experiments where the simulated traffic's bandwidth does not match theoretical. In the cases where there is a bandwidth mismatch and the explanation is not apparent, deeper exploration can be done in the PA graphical interface. An example of an IRE report can be seen in Figure 4 below.



Experiment ID	Master	Slave	Direction	Mean Simulated Band...	Mean Static Bandwidt...	Percentage Ratio of Si...	Min Simulated Latenc...	Mean Simulated Laten...	Max Simulated Latenc...
509	Sport0a	DMC0	read	6.24	6.25	-0.2%	352.00	487.52	651.20
509	SHARCXI1	DMC0	read	82.16	3,600.00	-97.7%	408.76	512.75	765.60
509	SHARCXI0	DMC0	read	82.16	3,600.00	-97.7%	415.80	508.86	664.40
509	CortexA5	DMC0	read	380.74	3,600.00	-89.4%	583.37	1,131.40	1,390.84
515	Sport3b	DMC0	read	6.24	6.25	-0.2%	607.20	758.56	880.00
515	Sport3a	DMC0	read	6.24	6.25	-0.2%	378.40	613.36	748.00

Figure 4 - IRE (IPA Results Explorer) Screenshot

Full automation flow: from specification to design to modeling

While the above flows are necessary to meet some of the automation goals listed above, a more cohesive flow is needed to meet the entire challenge. Several significant pieces of automation are still required; namely, populating AMBA Designer from a starting specification and PA platform (model) generation from the design. These will be referred to as AMBA Designer Automation and Platform Generation, respectively. Figure 5 below breaks out the entire automation flow

overview in one chart. To note in the figure; greyed in boxes indicate that the particular flow requires manual intervention for some reason. All other boxes indicate a portion of a separate automation flow.

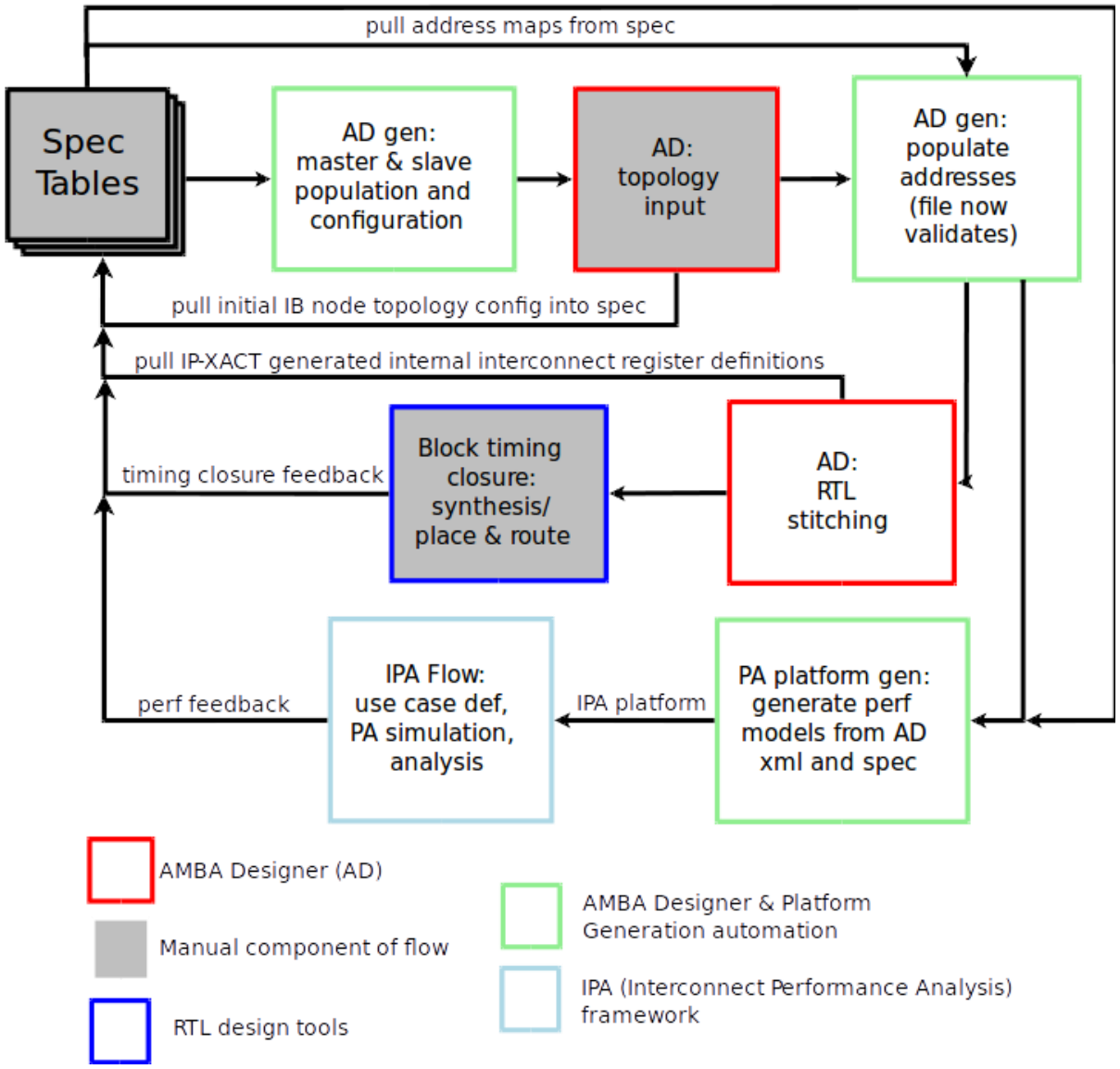


Figure 5 - Full Automation Flow

Specification

The format and method for storage of the specification are important decisions, as this has a direct implication on how specifications are maintained and shared. By using tables, we allow the specification to come from several sources: one excel workbook with many sheets, many separate CSV files, or a database storage backend. In any of these cases, automation can be added to populate specification data to any internal specification database. For our most recent design, we employed a combination of an excel workbook and automated population to a proprietary project specification and approval web application.

AMBA Designer (AD) Automation

As an AMBA Designer xml design is rather complicated, the automation flow is also somewhat complex. From Figure 5 above, the AD automation for a new design comprises three main steps:

1. Automated population of the masters and slaves, all configuration options for those nodes, and the master/slave connections ('architecture' mapping) from the specification.
2. Manual intervention in the AD GUI to create the topology, or hierarchy, of internal interconnect nodes which includes switches and interface bridges (IBs).
3. Automated address map population that matches the master/slave connections previously populated. AD requires physical connections in the hierarchy before address population is allowed.

Step #2 above could be avoided if the topology design were defined in the specification, for e.g., in a graph language representation. This was felt to be extremely tedious and hard to maintain as well as significant work; instead, the decision was made to keep the AD GUI as the primary source of input for the topology. While the GUI is used as input, additional 'helper' automation flows were defined to:

- Pull IB configuration back into the specification; this allows the timing closure, FIFO depths, and other settings to be defined in the specification without digging into sub-menus of the AD GUI.
- Extract the internal topology xml elements for a replay mode. This provides for population of a defined topology that can be applied to a design that may have evolved, but still has the same name and number of masters and slaves.
- Replay all steps of AD automation for specification updates by running step 1, populating a previously extracted topology and IB configurations from the specification as a replacement for step 2, and populating address maps in step 3.

It is the addition of these features which make the automation flow essentially single-sourced from one specification. Assuming that the masters and slaves are remaining the same in name and in quantity, any changes to the specification can be automatically pushed through steps 1-3 above in one replay automation step.

Finally, the AD RTL stitching proceeds via CLI (Command Line Interface) or GUI to generate the RTL directory structure. This directory is used to integrate with the rest of the SoC design as well as proceed to basic block level timing closure. During timing closure, any register slice feedback can be immediately applied to the specification, and the design regenerated as well as the specific settings shared with remote teams via the project's chosen revision / document control flow.

SoC Architecture Platform Generation for IPA and Platform Architect

To meet the requirement of a unified performance model, PA platform generation from the AD specification and XML is entirely automated. This is accomplished via the use of the Python code templating package Jinja2, which is traditionally used for web application templating. In this case, the templater uses pre-written and validated Tcl templates for particular master and slave models, clock and reset generators, and variable population. This results in the generation of a series of PA Tcl files, which are unified under a single, PA invocable, Tcl top file. The

automation invokes Platform Creator, which is the front end assembly GUI for PA, via the CLI which results in the appropriate PA XML platform.

One specific feature of this flow is PA variable management. As PA allows for sweeping across many different hardware configuration options in each SystemC model, there is a great deal of flexibility and design space exploration capability that is offered to the system architect. However, when hardware variables are defined, managing their configuration and default values is something that is typically done from within the Platform Creator GUI. The problem here is that now we have default values that must be maintained in the platform, if we don't explicitly have a hardware sweep definition in IPA that specifies their values. Platform generation automation allows for hardware values that are desired to be swept to be defined in the interconnect specification itself, maintaining a full automation flow to create the PA platform. This allows the user to use the automation, enabled via our platform generation automation and PA's Tcl automation APIs, to not need to manage platform variable values and thus have more than a single source of truth.

4. Results

The flows described in the methodology section have all, at one time or another, been applied to ADI's SoC design methodology. The final automation flow has most recently been successfully applied to a design process. Below, the table qualitatively compares our experiences with each flow.

Flow	**Model Accuracy**	**Operating Condition Coverage**	**Design Space Coverage**	**Schedule**
1. Static (spreadsheet) analysis	Lowest; theoretical maximum bandwidth with a derating value.	Is high.	Is high; can quickly change/permutate model variables	Takes minimal amount of time; simple spreadsheet or code manipulation.
2. RTL Simulation for interconnect performance analysis	Highest; simulating the actual design.	Is very low; potentially takes man-weeks to create a single use case.	Is very low; minimal, if not no design space coverage happens here.	Typically this flow takes so long that the scope of the work is reduced as the project progresses.
3. PA IMPO	Med-high; cycle-accurate interconnect with abstracted masters and slaves.	Is med; allows for fairly quick definition of specific traffic from masters; using IMPO automation, traffic traces are generated.	Med; allows for almost any configuration option in a given design to be swept. Takes time to set this up correctly.	Can take time to make the SoC platform model and configuration.

4. IPA	Med-high; same as PA.	High; baseline suites allow for automated simple cases, and complex use cases can be defined quickly via helper utilities. Many hundreds of results can be easily visualized via auto generated reports and plots. In the time it takes to define 10-20 use cases in PA IMPO, IPA can generate hundreds of use cases.	Med; same as PA.	Helps set up many use cases, some automatically via baseline, some defined at a high level with automatic permutations. Automatically generates sensitivity analysis visualizations.
5. Full automation flow: AD automation→ Platform gen→ IPA	Med-high; same as PA.	High; same as IPA.	Highest; allows for different specifications to automatically drive SoC platform model. Specification can also define hardware variables through both IPA and PA IMPO flows.	Helps to reduce the actual design time by single-sourcing specification, and does not require any time for design model creation. Still requires time for use case creation.

Table 1 - Comparing Methodologies

Static spreadsheet analysis to map application use cases to a design is something that should be done early on in the design cycle of any SoC. This is what drives the clock frequencies and data widths for the overall requirements of the project. However, this flow does not allow for tuning any further design parameters or finding design issues, other than obvious ones. On the opposite side of the spectrum, RTL simulation of a SoC can help tune every design parameter – but time constraints prohibit exploring design or use case space beyond more than one or two permutations. PA MCO brings a standards-based tooling environment, cycle accurate interconnect models and supporting master and slave models as well as high visibility into what is happening. Its Tcl automation APIs provide for automation around modeling, analysis and platform configuration. IPA leverages the PA flow and Tcl automation, adding the ability to define use cases at a higher level of abstraction, allowing for easier generation of many hundreds of permutations of simulation, along with automatic visualizations to aid in comparison across these permutations. While IPA along with PA IMPO are powerful modelling tools, they still require the creation of SoC platform models, which can be time consuming. The full automation flow automates the creation of these SoC platform models, along with the entire AMBA Designer design flow.

5. Conclusions

While all of the flows discussed in the methodology section add value in helping to create an interconnect that meets performance, only the final full automation flow allows for appropriate performance analysis of the interconnect while actively reducing the resourcing required to perform all steps in the flow. By reducing the work required to make architectural performance models of the interconnect as well as a single source of truth in a unified specification, any reconfiguration time is significantly reduced. The flow allows us to explore many more design candidates before settling on one to run through timing closure and RTL flows than would otherwise have been previously possible.

6. References

[1] ARM CoreLink AMBA Designer

[2] Synopsys Platform Architect MCO

[3] T. Ajamian. *SoC Architecture Analysis and Optimization Using Synopsys Platform Architect MCO*. SNUG Silicon Valley 2013.