SET TOP BOX SoC DESIGN METHODOLOGY at STMicroelectronics

Francois Remond, STMicroelectronics
Pierre Bricaud, Synopsys Inc

Abstract:

In this paper we will review how the IP Reuse SoC design methodology has evolved from its first introduction, heavily based on IP Reuse to a state-of- the-art design flow based on soft and hard IP block and floorplanning tools. This will be illustrated in one complex SoC present in the broadband communication market today, which is a Set Top Box IC containing a proprietary 64-bits processor and some general-purpose blocks, along with dedicated functions specifically designed by STMicroelectronics. In order to manage designs of this complexity, a top-down, block-based design style, relying on automatic floorplanning tools will be described. This design style is using the classical 'divide-and-conquer' strategy and is thus enabling a concurrent development process, guaranteeing timing convergence and correct chip assembly.

1. SoC/IP Reuse design methodology

The SoC/IP Reuse Revolution, as it was described by Gary Smith [2] in May 1997, was a first very strong media awareness of the design challenges that the designers would be facing. At that time, putting together IP blocks was associated to putting together Legos. Simple. The fundamental difference was that each Lego block was built to specifications, while IP blocks didn't have open standards or specifications. To support this new SoC/IP design methodology an industry consortium was created, called the Virtual Socket Interface Alliance (VSIA). Its goal was to create a set of standards for making IP reusable. These standards were defined and are well accepted today [3]. But these standards were defined independent of the application and integration design tools. At that time, this was obviously the best way to go. Rapidly, the integration of IPs' from different sources and different levels of perceived quality became a validation and verification nightmare.

In parallel, Synopsys and Mentor Graphics signed a Design Reuse Partnership agreement to support the new SoC/IP Reuse methodology. Result was the publishing of the Reuse Methodology Manual for SoC Designs [1] and a first shot at offering an IP quality self-rating spreadsheet called 'OpenMORE'. But let's go back to the problem of IP integration. The design of complex SoCs and derivatives for the telecom, wireless and consumer markets with the time to market constraints gave birth to what is known as platform-based SoC design methodology. Key to this design approach was the fact that the IPs' are described at a higher level of abstraction thus enabling a simple first approach to system integration. But whatever the design support media you choose, the fundamentals of SoC/IP Reuse must have been set. The example described is a pure result of the SoC/IP reuse methodology and illustrates very clearly that the engineering community has made significant progress to solving the time-to-market challenge.

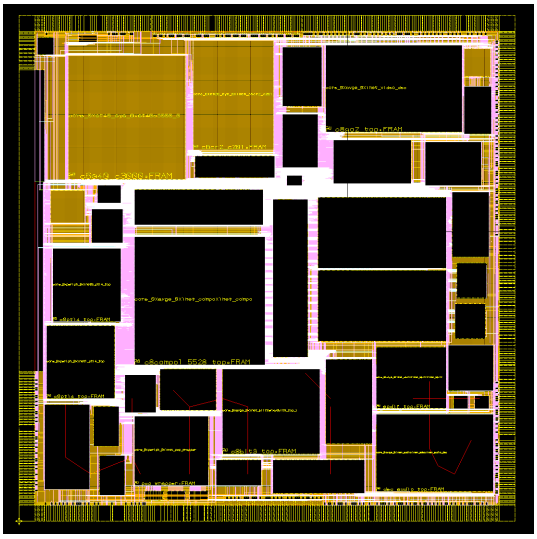
2. Design complexity and challenges

Today's circuits for Set Top Box applications belong to SoC design family. Chip complexity is in the range of 8 Million synthesizable gates and 3 Mbits of embedded static RAM memory, integration of analog IP's and 400 signal pads. Main clock frequency is 200MHz and circuits are organized around an internal proprietary bus. The challenge of such developments is clearly the time to market, which can be split into two phases: time to tape out, and time to production.

3. Circuit design flow overview

To achieve the time to market target, we have to deal with two objectives:

1. Shorten the time to tape out by taking advantage of design reuse techniques, enabling concurrent execution of front end and back end design and also insuring the predictability of design closure process.
2. Shorten the time to production by reducing - ultimately eliminating - the silicon chip iterations to solve bugs discovered during circuit evaluation solve bugs discovered during circuit evaluation.



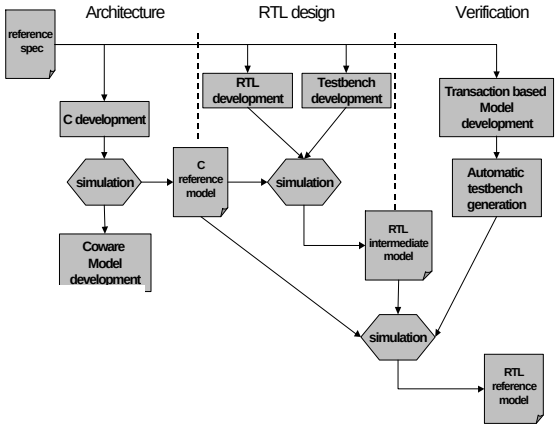


Figure 1. Front End Design Steps

Thus the design flow used in Set Top Box division is similar to the Spiral design flow described in Reuse Methodology Manual [1], with emphasis on functional verification techniques. It is composed of a front-end design flow for generic reusable IP's or specific blocks, including a physical prototyping step, and a SoC design and physical integration flow. Those steps are described in further details in the following paragraphs.

4. Front end IP and blocks design flow

Architecture study is mostly done through common 'C' simulations, although specific blocks (mixed digital-analog designs) require the development of Matlab models and tool chain. Tight loop between architects and RTL designers is needed during this step to insure that the resulting architecture makes sense in term of further RTL coding. Aiming to produce a reference 'C' model, the architects may also develop a model for HW/SW co-simulation using Coware's tool suite when the embedded software part is predominant for the block functionality.

RTL design is performed starting from the reference specification [figure 1]. The RTL code is simulated against the reference 'C' model to validate its functionality. At this point, RTL checker - HAL from Cadence Design Systems - is used to verify the correctness of the RTL versus the ST coding rules (Blue book guidelines). This has been proved to reduce problems when going to the synthesis step, enforcing compliance to best practice synthesis approach. Test bench development is carried on to simulate the RTL. DFT blocks (like BIST) are embedded in the RTL design, scan chain insertion is part of the synthesis process.

Synthesis scripts are developed using ACS based generic flow and templates, and in parallel detailed static timing analysis (STA) environment and target timing constraints are defined. DFT coverage is assessed at this point. Physical prototyping step then occurs through physical synthesis process (introduced starting .18um process to cope with VDSM effects on wire modeling). Physical prototyping is a key step to prove that the IP or specific block is valid for layout implementation, with all the information, set of timing exceptions, clock and high fan out nets description, needed to proceed further.
At the time of this project, the prototyping occurs "in context" of the final integration, which creates dependencies with the top-level design. The trend is to move to an "out of context" prototyping, which can occur early in the design process to improve the quality of the initial IP delivery. Nevertheless an obvious drawback of this evolution is the limited validity of such prototyping when several IP's will be grouped for the final integration. With the predominance of the interconnect in the timing closure, an IP which nicely converges in a $1mm^2$ area may be very difficult to close once merged with others in a $4mm^2$ box.

The resulting set of data is packed into a normalized front end package including RTL code, test benches and simulation environment, STA scripts, Physical synthesis scripts, and test coverage results.

These classical steps of front-end design have been complemented by exhaustive functional verification activity, based on automatic test benches generation system using Specman tool. It acts as quality insurance, to reduce the number of bugs discovered during silicon chip evaluation. To speed up this time consuming process, extensive usage of hardware emulation is required. As embedded software becomes an important

part of the IP functionality, this verification includes HW/SW co-simulation.

All those design steps are managed through a data management and bug-tracking system based on commercial products. Release of the final front-end package is done to an IP repository, having a cross-division wide visibility, or through a local repository for the limited usage of the targeted project.

5. SoC design and physical integration flow

At this point, the chip architecture study mainly focuses on embedded CPU and bus related topics like bandwidth requirement or detailed bus architecture; this last point having intimate links with physical floorplan of the chip. For example, for a bus using a multiplex strategy, the floorplan will dictate which blocks I/O's make sense to be muxed together, and thus drives the bus design.

5.1. Partitioning

It was chosen to apply a block-based top-down hierarchical design approach, where logical and physical partitioning are identical for the first level of hierarchy, and take advantage of the concurrency allowed by this choice between the top level design and block level design. The top-down block based method was chosen as opposed to the "virtually flat" approach, which allows less parallelism in the design execution. In this block based design approach, partitioning is done at RTL level, creating subsystems of reasonable size, up to 1 million synthesizable gates, with coherent functionality. Those subsystems may be composed of a collection of individual IP's which are merged at RTL level using wrappers. Consistent STA environment is created at the subsystem level by merging the IP level STA environments. This is a non-trivial design task because of the various clocking strategies, thus timing exceptions of the individual IP's.

For this circuit, initial "shopping list" was composed of 73 independent IPs' and functional blocks. Partitioning allowed to move down to 30 soft blocks and 13 hard blocks (mainly composed of CPU cores and analog IPs'). We foresee this grouping activity to be extended to a greater scale, allowed by the increased capacity of the back end tools. It is then the domain of the "virtually flat" design approach This is especially true for cost reduction versions of a circuit, while initial version still takes advantage of the concurrency allowed by a smaller granularity in the partitioning.

5.2. Top level physical and time budgeting.

In the top down design approach, timing and physical constraint are propagated from the top to the blocks through the physical and time budgeting activity. It starts once a top-level netlist of the block assembly is available. To build this netlist, it is not needed to have anything else but the pin definition of the blocks composing the circuit, and the specification of the top level interconnect. Along with this netlist creation, estimated block level models of size and target timings are needed to allow budgeting activity to start. Figure 2 gives an overview of the approach.

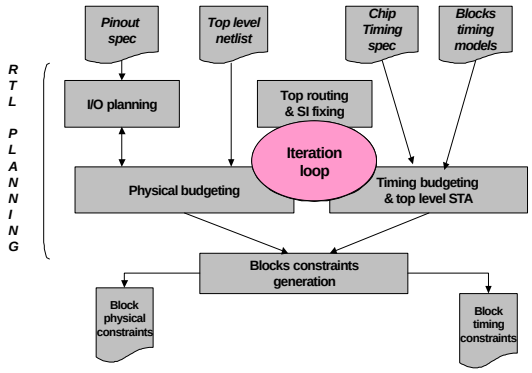


Figure 2. Design Planning Steps

Physical budgeting. Block size estimation is extrapolated from rough gate count obtained from low effort synthesis, or gate count available from previous implementation of those blocks in other circuit or process. Combined with technology target density, they will allow top-level floorplanning activity to proceed. Using IcWizard block based floorplanner, soft blocks are automatically shaped to reach an optimum density, keeping routing channels for inter block connectivity. This approach has been chosen, as opposed to connectivity by abutment and feedthrough propagation in the blocks, because it allows to perform top level routing during the floorplanning process, and thus allow to master rapidly the inter block connectivity delay. Once getting to the point of having top level floorplan executed with top level routing - including long nets buffering based on net length criteria - physical budgeting activity is completed and parasitic extraction takes place from the top level routed view.

It has been chosen to perform top level detailed routing at this step to take into account signal integrity issues at this point of the design process. As the crosstalk contribution can represent 30% of the total wiring delay in such process, it is mandatory to account for it for the top level routing, where channel based strategy is worsening the effect, forcing the co-linearity of the wires.

Timing budgeting. Target block level timing models, in the same way, are built either from the written specifications, using proprietary language to describe the timing arcs, then converted into STAMP model, or coming from previous implementation of those blocks - especially true when integrating IPs' widely used: USB's, PCIs', I2Cs'.

Using chip level timing constraints, first top level timing analysis run can be done, allowing first to debug the STA environment, then getting information about the possible timing problems. A first loop of timing budget refinements is done here, tracking the infamous 20ns negative slacks. Up to this point, this activity is independent from the physical budgeting. Once parasitic file is available from top level routing, it is used to perform the second time budgeting iteration. Timing issues highlighted there can be solved either by manual slack reallocation, or by floorplan optimization to speedup a critical path.

Iterations occur until a 'zero violation' STA is available; typically, but design dependent, we foresee two or three iterations.

Constraints propagation. Block level constraints are simply derived from the top-level design as a set of physical constraints (boundary, pin positioning, power grid) and timing constraints (SDC's) to allow final run of block implementation to occur. It must be noted that up to this point there is no dependency between top level and block level design activities, which can be conducted independently, by different teams in different locations.

5.3. Blocks implementation and chip assembly.

Block implementation is done using a generic automated flow relying on commercial tools like Physical Compiler and Apollo. During block implementation, emphasis has been put in signal integrity prevention and fixing, even though relying on a pessimistic static based approach. Timing convergence occurs through a one pass min-max optimization based on the clock skew budget. Clock tree is generated then and final timing adjustments are conducted.

As soon as a block is completed, it is incorporated in the top level design, through possible engineering change order (ECO) flow if timing or physical budgets are not met. In this case, related blocks budgets can be tuned to still maintain the final timing/physical closure.

At the end of this chip integration process, final STA and DRC/LVS are, by construction, free of violations.

6. Conclusion

We have tried to illustrate that SoC/IP Reuse design methodology is a reality. Complex chip designs are benefiting from the industry effort on IP standards. But we are still quite far from a homogenous design flow from a system level description to a GDS2 tape out, which today is splitted into three different signoff procedures: RTL, ASIC,GDS2. Some gaps have been underlined, one being the necessity to move from a golden C architectural model to the RTL design representation. On the other hand, new tools are appearing that allows a smooth transition between the RTL to the GDS2 design representation. The limitation of this hierarchical approach is the lack of flexibility in tackling the die size optimization, however the benefits of the described SoC/IP block based design flow is this scalable approach which opens the door to future complexity growth.

7. Acknowledgements

Authors would like to acknowledge very useful discussions with Thierry Bauchon, R&D Director at ST CMG STB division and to Nathalie Rocher for valuable contribution as technical writer and publishing of this paper.

8. References

[1] Michael Keating, Pierre Bricaud, Reuse Methodology Manual for System-on-a-Chip Designs, Third Edition, Kluwer Academic Publishers

[2] Gary Smith, 'The Revolution Isn't Coming-It's Already Here', VirtualChipDesign, May 1997

[3] VSIA, www.vsi.org