On Modeling and Simulating Chip Design Processes: The RS Model

Amir Hassine and Erich Barke

Abstract—The International Technology Roadmap for Semiconductors (ITRS) reports about the increasing divergence between what technological advances afford (in terms of the number of transistors on a single chip) and the capability to design these complex chips: the “Design Gap”. The chip design industry evolved into very sophisticated and complex processes needing managerial approaches to master them. Missing possibilities to evaluate course and outcome of projects in a simulative way entail soaring losses. In this paper, we present a pioneer approach that allows for modeling design processes in a formal manner. A prototype implementation of a simulator based on our model attests the appropriateness of the model.

Index Terms—Chip Design Process, Modeling, RS Model, Simulation.

I. INTRODUCTION

With the introduction of the first digital computers in the early nineteen-thirties, barely someone expected them to experience such a glamorous role for humankind. “I think there is a world market for maybe five computers” Thomas Watson, Chairman of IBM, said in 1943 and he wasn’t the only one to share this opinion. “There is no reason for any individual to have a computer in their home” annotated Ken Olson, President, Chairman and Founder of Digital Equipment Corp. in 1977. No one could expect that integrated circuits (ICs) could revolutionize computers and lead microelectronics into unimaginable dimensions: transistor size miniaturization down to nano-scales made the integration of up to billions of transistors on a single chip possible. Microelectronics has become an integral part of our daily life. Security, comfort, etc. and other branches owe their existence to the continuous progress in chip design.

The Federal Reserve Bank of Dallas refers to the immense contributions of the continuous advances in the semiconductor industry and its impact on the economy as the “microprocessor miracle”. The sustainable revenue growth of over 16% yearly since its beginnings in the 1970’s, turned the industry into a Wall Street darling. However, this trend began to slow down.

Because of aborted and cancelled projects, semiconductor companies sustain $2 to $4 billion losses yearly [1]. One of the main reasons for late-stage IC project cancellations are the missing approaches to simulate design processes in order to predict their courses and the resources needed.

Whereas lots of effort have been spent on defining and measuring productivity [2, 3], establishing business and technical KPIs [4], elaborating infrastructures for data collection [5] and analysis methods [6] and apart from few pioneer work [7, 8], formalization and simulation of design systems have been barely addressed. In times of immense cost and competition pressure, this turns out to be a prevalent issue.

In this contribution we present a novel approach, called Request Service Model (RS Model), which is appropriate for computer aided simulation due to its high-grade of formalism. The approach we present is inspired by Petri nets, but extends them to afford a realistic modeling and to adapt it to the facts and constraints of chip design. It delivers an instrument to represent design systems in a formalized manner and to allow computerized simulation. The RS Model allows for modeling design activities including allocated tools, computing resources, designers, costs, etc. Built into a simulator, potential bottlenecks can be recognized, what-if analysis can be carried out and critical factors such as deadline exceedance, resource overload or wastage can be identified in advance.

Section II overviews the chip design process and describes how to model it based on our approach. Section III presents a prototype implementation of a simulator based on the RS Model. Section IV concludes the paper and gives directions about future works.

II. THE REQUEST SERVICE MODEL: A STATIC VIEW

A chip design process does not differ much from other processes since it can be seen as the interaction of three constitutive elements: activities, design artifacts (DA) and resources. Activities build the skeleton of the process and transform design artifacts into different states. In the course of a process, design states are described steadily from higher to lower abstraction levels: the specification - typically in a human language, is transformed into a functional one. The latter is then synthesized into a structural description containing electronic components. Then, the structure is transformed into a physical description (layout) that can be taped out for manufacturing. Almost every activity is followed by a verification step to the check functionality’s correctness.
and constraints’ compliance.

The duration of an activity and the quality of its output (outgoing DA states) straightly depend on the quality and the complexity of the input (ingoing DA states) and on the properties of the used resources, e.g. designers’ experience, tool’s capability.

Similar to Petri nets, we use places (circles), tokens and transitions (rectangles) to graphically represent the elements of a chip design process. However, the RS Model comes up with multiple extensions that allow for modeling simultaneous resources shareability, assigning costs for the use of resources and monitoring outputs’ quality to control potential iterations. Most notably, the model allows for specifying behavioral models (so-called production/quality functions) to express the duration of an activity and the quality of its outputs in relation to the used input factors.

A. Modeling resources

In the context of the RS Model, resources are modeled as providers of services that are requested by activities. In the following we enumerate some resources and the services they offer:

- Designer: offers e.g. his ability to conduct activities and to use tools.
- Computing Resource: offers e.g. RAM and CPU.
- Tool: offers its appropriateness to execute activities and to automate their execution to certain levels.
- License: allows the use of certain tools, computing resources or libraries.

We represent each offered service through a token within the corresponding resource. The latter is represented by a place. By means of further attributes assigned to the tokens, the services (service properties) and the manner how they are delivered (behavioral properties) are specified. Fig. 1 shows the generic and a specific representation of a computing resource in the RS Model. As shown in Fig. 1, the service properties consist of:

- work power: denotes the quality and/or quantity of the provided service,
- availability: expresses the availability of the service in hours per day and
- cost: indicates the charges resulting from consuming a work power unit.

Service repartition and service intensity (behavioral properties) describe the dynamic aspects of the service delivery. In case of concurrent requests, service repartition decides about how the services are delivered to the competing activities. Requests may be served in chronological order (FCFS, First Come First Serve) as this is the case for RAM allocation, or, in other cases, may be split equally (∀=) or weighted (%).

In order to deal with the topics mentioned above and to model resources shareability, we introduce a special token called SimServices (SimS, see TABLE 1).

B. Modeling design artifacts

As shown in Fig. 2 and similar to resources, design artifacts are represented by places. The tokens residing in the place

<table>
<thead>
<tr>
<th>TABLE 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXEMPLARY SUMMARY OF A DESIGNER</td>
</tr>
<tr>
<td>Designer</td>
</tr>
<tr>
<td>work power</td>
</tr>
<tr>
<td>availability (h/day)</td>
</tr>
<tr>
<td>service intensity</td>
</tr>
<tr>
<td>service repartition</td>
</tr>
<tr>
<td>cost ($/h)</td>
</tr>
</tbody>
</table>

*Designer’s experience in carrying out/using particular activities/tools.
stand for the different states that the DA undergoes during the design process. Each state (e.g. specification, HDL\textsuperscript{1}-code, placed netlist) has a quality and a format attribute. The latter denotes the specific format in which the state is materialized (e.g. text document, Verilog, DEF\textsuperscript{2}). Each design artifact is characterized by a complexity level. Defining and measuring quality and complexity of chip designs are addressed within the project PRODUKTIV+ [9] as well as in [2] [3].

![Design Artifact](image1)
![Design Artifact USB complexity = 0.8](image2)

**Fig. 2:** Representation of a design artifact in the RS Model

### C. Modeling activities

Activities represent the transformative part of a design process. They transform DA states and therefore request services from resources. The effectively granted services and the properties of the input states (i.e. quality and complexity) decide on their duration and the quality of their output. The functionality of an activity is built up of three blocks (Fig. 3):

- **Pre-Guards:** check the existence of the required input DA states and whether the granted services meet the minimum required.
- **Code Region:** contains two mathematical expressions. A production and a quality function correspondingly express the quantity (per time unit) and the quality of the produced output states, dependent on the production factors (i.e. granted services and quality/complexity of input DA states).
- **Post-Guards:** monitor the quality of the output and control the initiation of loops and iterations.

![Activity](image3)

**Fig. 3:** Representation of an activity in the RS Model

The requests of an activity consist of MinReq - the absolutely minimum services needed to start an activity - and OptReq: the concrete services requested dependent on the actual situation. For example for a larger design, more RAM should be requested.

### III. The Request Service Model: Towards Simulation

In Section II we presented the RS Model form a static point of view. The introduced techniques may be used to model a planned process (i.e. sequence of activities to be carried out and resource allocation) or snapshot an actual state of a running - or simulated - process (i.e. current running activities, resources’ load, produced DA states, etc.).

To pass from statically modeling to simulating processes, the - static - model has to be extended by a control mechanism to coordinate the communication between activities (services consumer) and resources (services provider). Furthermore, it is necessary to provide adequate interfaces to enter the process to be simulated, to codify the activities’ behavioral models and to log the simulation’s progress in a flexible, extensible and machine-readable manner.

Adrenaline (Fig. 4) is the name of the prototype simulator we implemented based on the RS Model [10].

![Snapshot of the simulator prototype](image4)

**Fig. 4:** Snapshot of the simulator prototype

Adrenaline mainly serves as a proof of concept to evaluate the suitability of the RS Model to represent and simulate chip design processes. Concerning the activities’ behavioral models, we provide an approach to specify them but we use placeholder functions in the current implementation. Realistic functions are being investigated. In the current implementation, Adrenaline is intended to simulate predefined processes: it delivers the duration of the process and the quality of its outcome for a predefined resources allocation.

---

\textsuperscript{1} Hardware Description Language, e.g. Verilog or VHDL.

\textsuperscript{2} Design Exchange Format
A. Process entry and log

Because of its flexibility and standardization, we chose XML to specify and read in the process into the simulator. In addition, we implemented a graphical facility to comfortably enter the process to be simulated per drag-and-drop (Fig. 5).

The tolerance allowed in arranging the process elements (activities, DAs and resources) is codified in OWL. Using ontologies allows for representing concepts within a domain - in our case chip design - and the relationships between them. By doing so, we separate the generic knowledge (general configuration of a chip design process) from the specific application (a specific process to design a specific DA). This leads to a minimum risk that the user enters a corrupt semantic, since he can only build the semantic allowed by the ontology.

An automated procedure translates the graphical entry into an RS Model, see Fig. 4.

The simulation progress is logged in XML. Resources’ loads, accruing costs, etc. are recorded at every simulation step to afford easy reconstruction of the simulation results.

B. Behavioral models

The simulation results’ quality depends to a great extent on the activities’ behavioral models, i.e. production and quality functions. The present prototype simulator uses placeholder functions but implements the possibility to specify, interpret and apply such functions. Determining the latter is a continuous process that should be constantly updated. Therefore, the implemented mechanism must feature the necessary flexibility and extensibility to allow for entering, updating and ex post editing functions without changing the simulator code.

To express formulas, we combine the selector point notation - as known from object oriented programming - and the polish notation - first operators and then the operands are specified. In this vein, we simplify addressing process elements as well as nesting and interpretation of formulas. As an example, suppose that the quality of an output token is the product of the designer’s (D) experience and the quality of the DA’s (DAx) input state (token HDL). This will be codified as:

\[ (*) \text{DA}_x.\text{HDL}.\text{quality}, \text{D.E} \]

Requests and iterations’ conditions are expressed in a similar manner. A (min) request to acquire a designer D with a minimal experience of at least 0.7 (somehow defined, e.g. [11]) and a computing resource C with at least 0.5 GB RAM available, may be expressed as:

\[ \& (\geq, \text{D.E}, \text{workpower}, 0.7), \]
\[ (\geq, \text{C.RAM.workpower}, 0.5), \]
\[ (\geq, \text{D.SimServices.workpower}, 1), \]
\[ (\geq, \text{C.SimServices.workpower}, 1) \]

In case of allocation unambiguity, resources can be addressed by their generic types: e.g. Designer.SimSercices instead of D.SimServices.

C. Activities-Resources communication

Until now, we assumed that activities send their requests directly to the resources. As long as requests are not coordinated by a central instance, deadlocks may occur. As an example, consider two activities requesting a designer and 1 GB RAM (MinReq). We assume that the designer can carry out at most one activity simultaneously and that the computing resource offers 1.5 GB in total. If the computing resource grants memory to the first activity and the designer is assigned to the second, both activities would lock mutually: each one waits for the missing resource to be released.

To deal with such problems, we introduce a special element: the R Matrix. Activities’ requests are collected by the latter. It watches resources loads and ensures a deadlock free assignment of services.

Since the current prototype is based on the assumption of predefined activities-resources allocation, the role of the R Matrix is restricted to forwarding the requests to the resources in a manner that avoids deadlocks. However, we plan to use it to offer the opportunity to carry out allocation optimization by claiming services from a pool of resources instead of a specific one. A proper algorithm employed by the R Matrix, could then pickup the optimal resource from the pool.

D. Simulation run

The basic simulation run is outlined in Fig. 6. The controller is concerned with the coordination of the simulation steps and the control of the data flow between the process elements.

\[ \text{Web Ontology Language} \]
A simulation step is thereby broken down into four phases. Depending on the phase signaled by the controller, particular elements respond and execute particular procedures:

- **Phase 1 - Check (C):** Pre-Guards check the availability of the needed input DA states (1.1).
- **Phase 2 - Request (R):** Transitions send out their requests to the R Matrix (2.1).
- **Phase 3 - Grant (G):** The R Matrix checks the availability of the MinReqs (3.1). In the affirmative case, the R Matrix forwards the OptReqs (3.2). Taking into consideration the resources/services properties (e.g. shareability, service intensity), the resources grant thereon the services to the transitions (3.3) and the R Matrix notifies the activities about the granted services (3.4).
- **Phase 4 - Produce (P):** Active transitions (continue to) produce a portion of the output. If after this substep, production is completed, the Post-Guards check the outputs’ quality and initiate potential loops (4.1) or create a corresponding token of the DA (4.2).

### IV. CONCLUSION

Mastering the complexity to which chip design process nowadays evolved, requires the elaboration of adequate mechanisms to model and simulate it. On this note, PRODUKTIV+ delivers pioneer contributions.

In this work, we presented an innovative framework to model and predict the duration and the quality of the output of chip design processes in a simulative manner. We endowed the approach with the necessary formalism, which allows for a computer-aided simulation.

A prototype implementation of a simulator based on the RS Model has proven the suitability and the ease of use of the approach we developed. Presently, we have implemented placeholders for activities’ behavioral models. The latter are subject of ongoing investigations. We have defined a flexible and extensible methodology to codify them.

The close collaboration with large semiconductor companies within PRODUKTIV+ assures a solid basis for further verification and enhancements of our approach and the determination of realistic activities’ behavioral models.

### REFERENCES


