44. R. E. Bank, *PLTMG: A Software Package for Solving Elliptic Partial Differential Equations. User's Guide 7.0,* Philadelphia: SIAM, 1994.
45. A. Brandt, Algebraic multigrid theory: The symmetric case, in *Proc. Int. Multigrid Conf.,* Copper Mountain, CO, 1983.
46. J. W. Ruge and K. Stüben, Algebraic multigrid (AMG). In S. F. McCormick (ed.), *Multigrid Methods,* Vol. 5 of Frontiers in Applied Mathematics, Philadelphia: SIAM, 1986.
47. C. P. Ho et al., VLSI process modeling—SUPREM III, *IEEE Trans. Electron Devices,* **30**: 1438–1453, 1983.
48. M. E. Law and R. W. Dutton, Verification of analytic point defect models using SUPREM-IV, *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.,* **7**: 181–190, 1988.
49. G. Hobler, P. Pichler, and K. Wimmer, *PROMIS 1.6: User's Guide,* Tech. Rep., Vienna, Austria: Technical University, 1991.
50. M. G. Hackenberg et al., Coupled simulation of oxidation and diffusion in VLSI wafer fabrication. In A. Sydow (ed.), *Proceedings of the 15th World Congress on Scientific Computing, Modelling and Applied Mathematics—IMACS,* Berlin: Wissenschaft und Technik Verlag, 1997, Vol. 3, pp. 587–592.
51. S. W. Director, W. Maly, and A. J. Strojwas, *VLSI Design for Manufacturing: Yield Enhancement,* Boston: Kluwer Academic Publishers, 1990.
52. S. R. Nassif, A. J. Strojwas, and S. W. Director, FABRICS II: A statistically based IC fabrication process simulator, *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.,* **CAD-3**: 40–46, 1984.
53. F. Fasching, S. Halama, and S. Selberherr (eds.), *Technology CAD Systems,* Wien: Springer-Verlag, 1993.
54. O. A. McBryan et al., Multigrid methods on parallel computers—a survey of recent developments. *Impact Comput. Sci. Eng.,* **3**: pp. 1–75, 1991.
55. D. W. Yergeau, R. W. Dutton, and R. J. G. Goossens, A general OO-PDE solver for TCAD applications. Paper presented at 2nd Annu. Object-Oriented Numer. Conf., Sunriver OR, 1994.

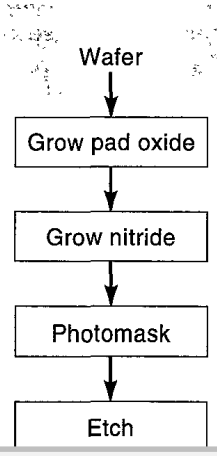
WOLFGANG JOPPICH
German National Research Center
for Information Technology
SLOBODAN MIJALKOVIĆ
University of Niš

SEMICONDUCTOR PROCESS REPRESENTATION

Semiconductor chip manufacturers, foundries, research laboratories, and other enterprises all use some sort of representation of the semiconductor fabrication process in order to make or design semiconductor devices and integrated circuits (IC). In their most elementary form, such representations may be textual or graphical and intended solely for human interpretation. Of much greater use, however, are highly structured or formalized representations that can be understood and manipulated by a collection of computer programs. The purpose of such a process flow representation is to capture key information for one or more purposes in the life of fabrication equipment or operators, to design knowledge about a process under development or optimization. Process representation is of particular importance in the semiconductor field because of the process-intensive nature of semiconductors. That is, the key characteristics of semiconductor products are highly dependent on the specific details of the process used to manufacture them.

There are many ways to describe or document a semiconductor process, and the process flow representation can be variously thought of as a language (if it has a textual form), a data structure, or, if sufficiently powerful and comprehensive, a knowledge base. Initially, when little is known about a process, an overview with many details hidden or unstated is desirable. This allows a big picture of the process so that at least the intent of the process can be understood. A circuit or device designer will generally be concerned with the various material layers, how they are patterned (what masks to specify), what regions are implanted with dopants, and so on. Physical realization of the process requires synthesis of a process flow to achieve the designers' intent and typically involves computer simulation of key process steps. A presentation of the process similar to a programming flowchart shows the main flow of control. All of the detailed exceptions, such as what happens when something out of the ordinary occurs, are hidden. Figure 1 shows the initial sequence of steps of a hypothetical but typical process. The process starts with a silicon wafer of known characteristics, and a pad oxide is grown followed by a nitride growth or deposition. A photomask step is used to pattern a protective resist layer on the wafer so that the subsequent etch step will selectively remove the nitride on specific areas of the wafer. Actual fabrication will generally require expansion of this simplified process flow and provide details in both sequence structure (substeps) and equipment-specific processing parameters. In the typical nitridation step, for example, the wafer is first cleaned and then the nitride material is deposited, using a particular schedule of gas flows and temperatures, often called a recipe, in a particular furnace. Afterward, the thickness of the deposited nitride may be measured as a standard part of the complete nitridation step.

In a real factory, there are, of course, other details that are important, some of which are often not written down. Such implicit details may be part of the knowledge, experience, and training of the fabrication operators, the equipment specialist



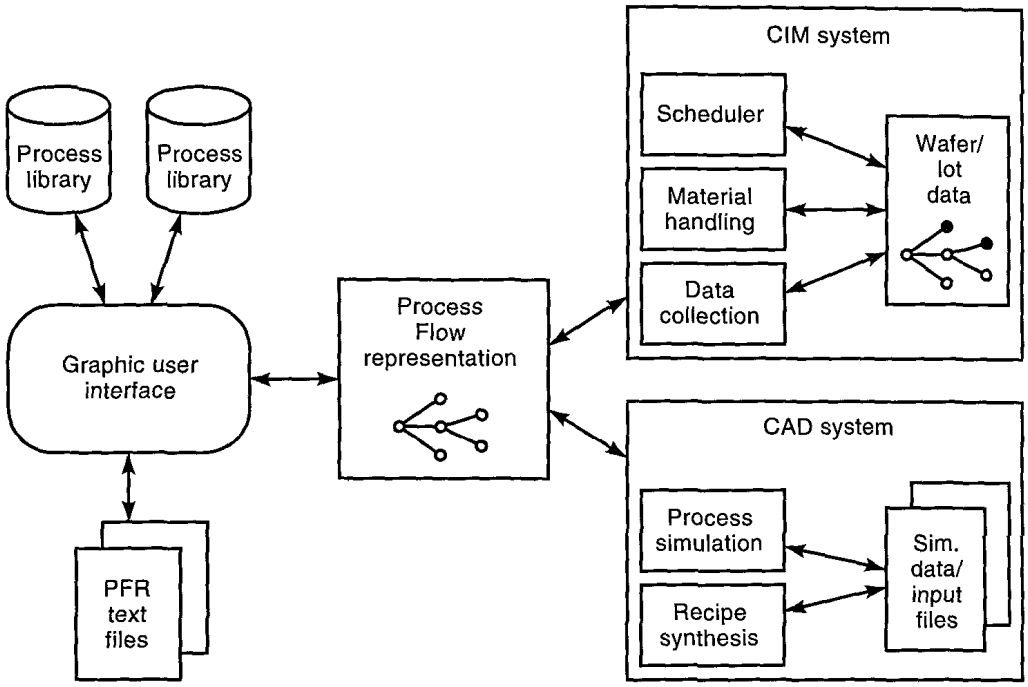


Figure 2. A unified process representation provides a common interface to various applications. A process representation may be created by a user through a combination of graphical user interfaces or editors operating on textual process descriptions and may draw process steps from one or more process libraries that could reside either locally or be accessed via a computer network. The process representation may be utilized or integrated with applications supporting fabrication or simulation.

or engineer, or of the equipment developer and manufacturer. The details may be embodied in multiple places, so that which details are used depends on when and where the ICs are made.

With computer representations for the process flow at various levels of detail, software programs that use process information may perform, for example,

- simulation
- safety checks
- instruction formatting
- data collection
- data reduction
- control
- process analysis and diagnosis
- scheduling
- rule-based or intent-based process synthesis

A complete software process flow representation system consists of four basic elements: the information model, user and programmatic interfaces, a base collection or library of processes, and a set of application programs that use or manage the process representation, as shown in Fig. 2. These elements enable the process representation to act as a general-purpose, unified means for expressing what is already known about the process, or what is learned about a process in the course of design, simulation, or manufacturing itself. A unified process representation is one in which the knowledge about the process is represented coherently and in a uniform fashion in order to bridge and integrate related activities (e.g., process design and manufacture). A unified process representation organizes the various levels of detail that are essential to the making of an IC and provides a comprehensive framework for knowledge about process steps.

describing both process structure (e.g., linear sequences of process steps or hierarchical decompositions of complex processes) and the organization of process details (e.g., details about what happens to a wafer during a process as it is subjected to specific gas flows, thermal treatments, etc.). Examples of the information expressed by a process representation include the process structure, control structure, simulation results, desired effect on the wafer, processing models, equipment operation or microprograms, scheduling data, and testing and yield results. Once a process representation is structured around such a model, application programs can be written to accomplish particular tasks. For example, a process representation may be used to generate fabrication instructions for people or machines—that is, the representation can be viewed as a program that operates on wafers as inputs, transforms them, and produces wafers as outputs. Alternatively, the representation can be viewed as data that include knowledge as to specifications, documentation, and the machines or other resources required to manufacture the product IC; software programs then use these data to accomplish fabrication or other tasks.

The second key element of a process representation system is the mechanism or interface for capturing, storing, and accessing process information. One simple approach is to write a process flow in a prespecified textual format or language, which is then read and interpreted by computer. Graphical user interfaces (GUI) are generally preferred by users who are not programmers; the GUI helps guide an engineer or designer in the creation and modification of process steps and the assembly of these steps into correct process flows. In addition to human interfaces, well-defined and standardized application program interfaces are critical to enabling a multitude of manufacturing or computer-aided design (CAD) systems to use and manipulate process information.

The third element of a working process representation sys-

steps and reusable subprocess modules that can be integrated by a process designer to create new complete manufacturing processes.

Finally, a process flow representation is of limited value without a supporting set of computer integrated manufacturing (CIM) or CAD applications to enable the accomplishment of actual fabrication or design goals.

In the following sections, process flow information models, as well as user and application program interfaces and process libraries, are discussed in detail. These form the generic core of a process representation system. More advanced issues and the current state of the art regarding the integration of process representations with CAD and CIM applications, as well as industry standardization efforts, are then described.

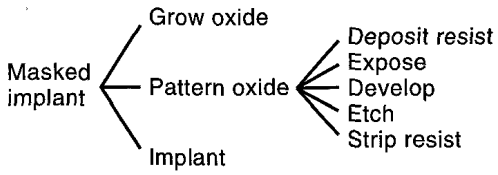
INFORMATION MODEL

The information model is fundamental to creating a shared understanding of terms and definitions in a formal process representation. At its most basic, a semiconductor fabrication process can be thought of as the movement of a silicon wafer through a series of process operations or steps, each step occurring in a piece of equipment where the wafer is subjected to some treatment that causes some desired change in the wafer (e.g., addition or removal of thin film layers, changes to material conductivity). The information model details both what is meant by *process sequence* and what can be said about what happens during each process step.

Process Sequence

Complete IC manufacturing processes are frequently thought of as being divided into smaller sequences of steps or modules (e.g., well formation, active area definition, metalization). Therefore, a fundamental "chunking" abstraction capability of process representations is the ability to describe a manufacturing process as composed of sequences of process building blocks or components.

Because each component may itself have subcomponents (e.g., subprocess steps), process flows are typically represented in a hierarchical or tree structure, as illustrated in Fig. 3. This hierarchical decomposition also enables modular process development, as the same process step (e.g., clean steps, thin oxidations, resist development) is often used at several points in the same process flow or across multiple process flows. In some process representations, the number of levels (and terminology at each level) is predefined (e.g., a process flow consists of process modules made up of unit process steps, which each occur on a specific piece of equipment). Such fixed hierarchies have been found to help communities of users structure and share complex processes. On the other hand, many process representations do not impose a strict hierarchy and provide for arbitrary levels of process sequencing (e.g., each process step can be decomposed into smaller process steps as needed to describe the process to whatever detail is desired). For example, while a unit process step might be a thermal operation in a furnace, it is often desirable to break this into smaller sequences of time blocks or events where temperatures or gas flows are changed during the process.



Several process representations have also been proposed that deal explicitly with more sophisticated process sequencing requirements. One example is timing constraints on the execution of process steps. It is often critical that one process step be immediately followed with zero or finite delay by the next step (e.g., polysilicon deposition after a gate oxidation), and attributes on a process step have been used to express such requirements (e.g., tagging a process to indicate that all of its substeps must be done within specified allowable delays). Process sequences may also be conditional on certain states or other events. The most common case is rework loops. Programming languagelike constructs can be used to specify under what conditions a change in the normal process sequence is required (e.g., to remove resist after a failed inspection followed by reinsertion in a photolithography step). In process representations that seek to support experimental splits or sophisticated feedforward capability, additional branching, looping, and other process sequencing capability for individual wafers in a lot (or for splitting or merging/batching wafers and lots) is also provided.

Generic Process Model

In addition to process sequence information, details about individual process steps are needed. The second key idea in a process representation is that specific information can be associated with process steps at various points in the process hierarchy; this information is usually captured in the form of attributes of the process step that express detailed or aggregate information at the appropriate point in the process. An example of a scheduling-related attribute is the time required to perform an operation, where this time might be the sum of the time-required attributes associated with the process step's subcomponents. To help organize and structure detailed unit process information, a generic model for semiconductor processing, as conceptually pictured in Fig. 4, has been defined. The process representation then supports the specification of desired states of the wafer, environment, or equipment at various points during fabrication as dictated by this generic process model.

During a process step, a wafer (or several wafers) is contained within some physical environment that has been generated as a result of settings on a fabrication machine within a facility. These settings are, in turn, controlled or dictated by a program or recipe. The layering in Fig. 4 indicates a number of boundaries: between the wafer and the wafer environment, between the wafer environment and machine/facility, between the machine/facility and settings (as well as readings), and finally between settings/readings and control programs. This conceptual layering is loosely guided by the

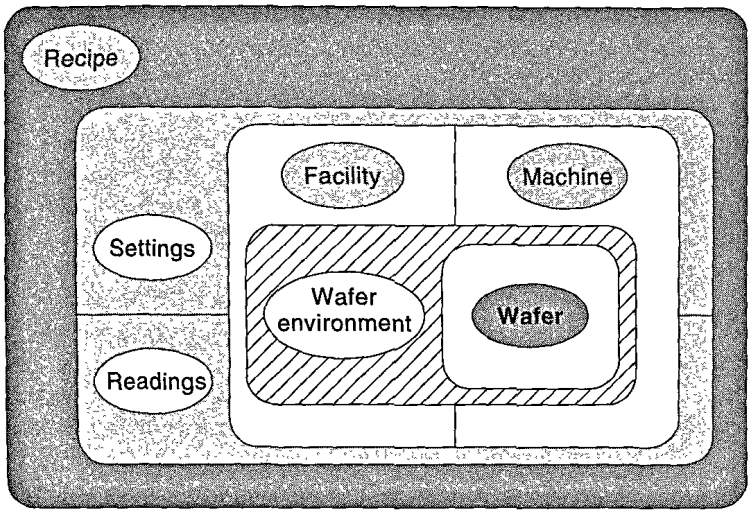


Figure 4. A conceptual model for semiconductor fabrication, identifying groups or categories of state information and interfaces between those states as they occur during IC manufacture.

shown in Fig. 4 or through a chain of such interfaces. Each may evolve over time due to internal interactions or as it is affected through interaction with the surrounding or enclosed entities.

This partial decoupling of entities (or the states of those entities) motivates a generic model of the semiconductor process to enable identification and differentiation among categories of state information corresponding to the partitioning shown in Fig. 4. In general, a state description may be specified directly (e.g., to indicate the desired or resulting state) or indirectly (e.g., as a delta or change in state that the step is intended to accomplish).

Wafer State. Of key interest is the state of the wafer at the completion of the process as well as at intermediate points in the process flow. While potentially infinite in complexity and detail, the process representation typically captures only those aspects of the wafer state that are necessary for either further processing (e.g., states that indicate what materials are on the surface of the wafer that enable safety or design rule checks) or further modeling (e.g., representations of individual devices to sufficient detail that desired process and device simulation can be performed). Common descriptions of the starting material include crystal orientation, resistivity, and carrier type of a wafer. Other state descriptions may include surface topography, bulk dopant concentrations, thin film stresses, and other geometric and parametric properties of the wafer. A typical desired change in wafer state is the addition or removal of a thin film of specified thickness or properties (e.g., deposit a 0.5 μm silicon dioxide layer). This is also sometimes termed the *effect* that a process has (or is desired to have) on a wafer.

Wafer Environment or Treatment. The wafer environment captures the relevant physical environment that the wafer is subjected to during processing. This treatment can be described as functions in position and time of temperature, par-

Machine/Facility. The machine state might include the current machine setup and configurations during operation, such as valve positions or the voltages across plates in plasma equipment. The machine resides within a facility that has attributes such as gases, airborne contaminants, and staff.

Settings and Readings. Settings correspond to the desired positions of knobs or other controls and may vary discretely or continuously as a function of time in response to operator or automated instructions. Examples of readings are the current shown on a meter of an ion implanter and a temperature derived from a furnace thermocouple.

Implementing the Information Model

A great deal of progress has been made in identifying a generic process model for unit process steps, as well as generic process sequencing mechanisms. Because of the complexity and varying scope of the problem domain, however, modern process representation implementations use process modeling and representation techniques that are extensible (that is, capable of easily being extended to accommodate new kinds of knowledge about processes for new and different purposes).

Process representation implementations may be divided into three basic types: programming language based, knowledge based, or hybrid systems. In the programming-language-based approaches, such as FABLE (developed at Stanford) and BPFL (developed at UC Berkeley), the process is explicitly represented as a program in a specialized programming language to be executed. In knowledge-based approaches, the process representation is treated as a general knowledge representation problem; the Stanford MKS and PDS systems are examples of this approach. The hybrid approaches attempt to combine the benefits of the other two. The MIT PFR is an example of the hybrid type; a textual form can be used to specify processes (or the same textual form can be used as an interchange format to exchange process information between different systems). A sample of this flow representation is shown in Fig. 5. Other systems have also adopted a hybrid approach; the Texas Instruments MMST system, for example, adopts a language-based front end with an object-based back end or application programming interface. These distinctions are not sharp; for example, a knowledge-based approach may also include mechanisms for representing control flow by means of explicit computation in an embedded programming language. These implementation approaches are closely related to the human and programmatic interfaces they utilize.

USER AND PROGRAM INTERFACES

A number of possible interfaces, both for human interaction and computer program access, can be utilized for the capture and expression of specific process flow and unit process step information. First discussed are methods for human interfaces to the process flow, followed by issues in computer-accessible representation.

Human Interfaces

DOCKET ALARM

Explore Litigation Insights

Docket Alarm provides insights to develop a more informed litigation strategy and the peace of mind of knowing you're on top of things.

Real-Time Litigation Alerts

Keep your litigation team up-to-date with **real-time alerts** and advanced team management tools built for the enterprise, all while greatly reducing PACER spend.

Our comprehensive service means we can handle Federal, State, and Administrative courts across the country.

Advanced Docket Research

With over 230 million records, Docket Alarm's cloud-native docket research platform finds what other services can't. Coverage includes Federal, State, plus PTAB, TTAB, ITC and NLRB decisions, all in one place.

Identify arguments that have been successful in the past with full text, pinpoint searching. Link to case law cited within any court document via Fastcase.

Analytics At Your Fingertips

Learn what happened the last time a particular judge, opposing counsel or company faced cases similar to yours.

Advanced out-of-the-box PTAB and TTAB analytics are always at your fingertips.

API

Docket Alarm offers a powerful API (application programming interface) to developers that want to integrate case filings into their apps.

LAW FIRMS
Build custom dashboards for your attorneys and clients with live data direct from the court.

Automate many repetitive legal tasks like conflict checks, document management, and marketing.

FINANCIAL INSTITUTIONS
Litigation and bankruptcy checks for companies and debtors.

E-DISCOVERY AND LEGAL VENDORS
Sync your system to PACER to automate legal marketing.