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 a fabrication process or semiconductor product, from early conceptualization through design and manufacture. Such information ranges from manufacturing details, including instructions for 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

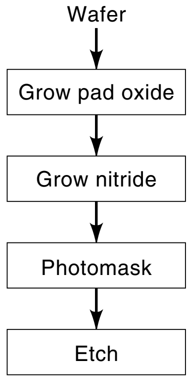

Figure 1. Semiconductor process representation involves multiple levels of detail. Shown here is a simplified sequential process flow for the selective creation of active areas where transistors will be formed.

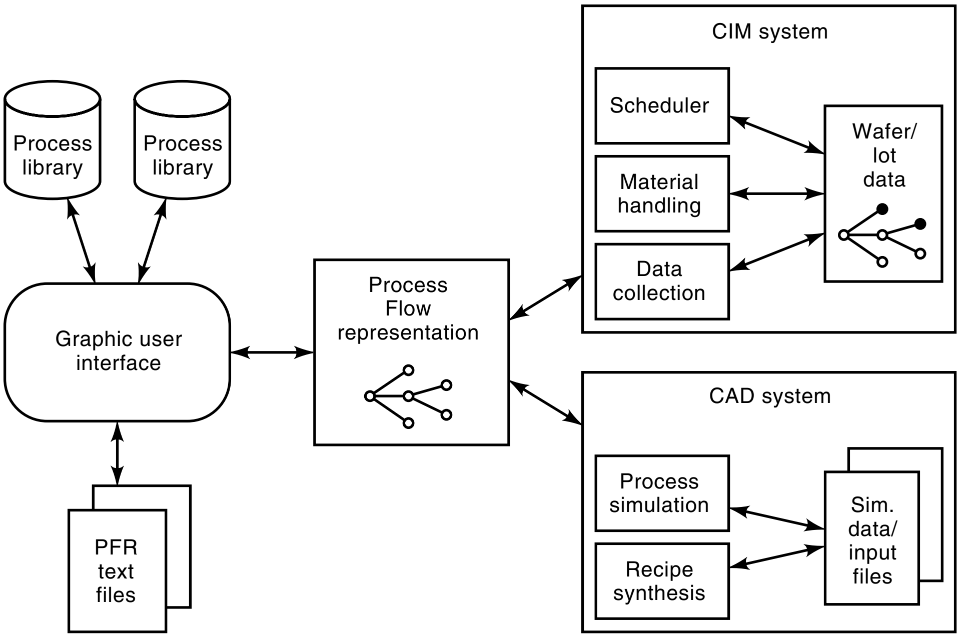

Figure 2. A unified process representation provides a common interface to various applications. A process representation may be created by a user through a combination of graphical user interfaces or editors operating on textual process descriptions and may draw process steps from one or more process 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.

The first element is an information model that specifies the type and format of process information that can be expressed. In particular, the information model defines the methods for 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 system is a collection of process libraries, which provide the available processes for one or more particular fabrication facilities (fabs). Such libraries may also include unit process 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.

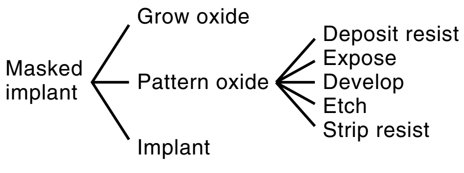

Figure 3. A hierarchical or tree decomposition for a masked implant process step. Each substep can be decomposed into an ordered sequence of smaller substeps.

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 physical containment that exists during the processing (i.e., wafers within wafer environments within machines). One of these can affect another only through one of the boundaries

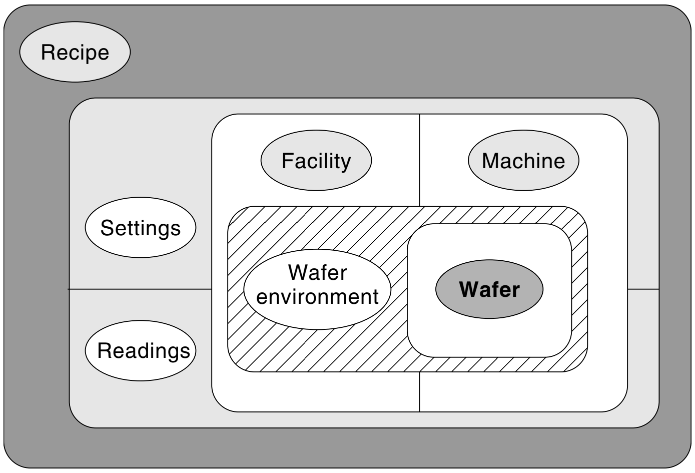

Figure 4. A conceptual model for semiconductor fabrication, identifying groups or categories of state information and interfaces between those states as they occur during IC manufacture.

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, partial pressures of ambient gases, and so on. These parameters are typically thermodynamically intensive, a property that helps to distinguish them from machine state parameters.

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

Two different approaches (or a hybrid of these approaches) have been widely used to enable engineers and operators to specify process steps and flows. On one end of the spectrum

```
(define cmos-baseline
  (flow
   (:documentation "CMOS Baseline Process")
   (:body
    initial-epi
    well-formation
    active-area-definition
    field-formation
    channel-formation
    source-drain-definition
    bpsg-passivation
    contact-definition
    metal-definition))))

(define well-formation
  (flow
   (:body n-well-formation
          p-well-formation)))

(define n-well-formation
  (flow
   (:body stress-relief-oxide
          lpcvd-silicon-nitride
          n-well-pattern
          nitride-plasma-etch
          n-well-ion-implant
          resist-ash)))

(define p-well-formation
  (flow
   (:body
    n-well-cover-oxide
    nitride-wet-etch
    p-well-ion-implant
    well-drive
    well-oxide-wet-etch)))

(define active-area-definition
  (flow
   (:body stress-relief-oxide
          lpcvd-silicon-nitride
          active-area-pattern
          nitride-plasma-etch
          p-field-pattern
          p-field-ion-implant
          resist-ash)))

(define field-formation
  (flow
   (:body n-field-pattern
          n-field-ion-implant
          resist-ash
          field-oxide
          nitride-wet-etch)))

(define channel-formation
  (flow
   (:body stress-relief-oxide-wet-etch
          dummy-gate-oxide
          n-channel-definition
          p-channel-definition
          dummy-gate-wet-etch
          gate)))
```

```
(define n-well-cover-oxide
  (flow
   (:documentation
    "Grows a thick cover oxide using a thermal treatment")
   (:permissible-delay :minimal)
   rca-clean
   (operation
    (:body
     (:change-wafer-state
      (:oxidation :thickness (:angstroms (:mean 5100 :range 250))))
     (:treatment
      (thermal-rampup-treatment :final-temperature 950)
      (thermal-dryox-treatment :temperature 950 :time (:minutes 30))
      (thermal-wetox-treatment :temperature 950 :time (:minutes 175))
      (thermal-dryox-treatment :temperature 950 :time (:minutes 30))
      (thermal-rampdown-treatment :start-temperature 950))
     (:machine Thick-Oxidation-Tube)
     (:settings :recipe 240)))))


(define (thermal-rampup-treatment final-temperature)
  (sequence
   (:thermal :temperature 800
             :time (:minutes 20) :ambient :N2)
   (:thermal :temperature 800
             :time (:minutes 10) :ambient :N2)
   (:thermal :temperature 800 :ambient :N2
             :time (:minutes (/ (- final-temperature 800) 10.0))
             :temp-rate 10)
   (:thermal :temperature final-temperature
             :time (:minutes 10) :ambient :N2)))
```

Figure 5. Sample process flow representation. Here a textual programming language format is used to capture key information about a process flow.

lies the programming language analogy and approach, with engineers editing text files that express process steps as somewhat like subroutines or procedures, with the ability to identify variables and parameters in those steps to enable generalization or reuse of those steps in different process flows. From the very earliest work on programming languages for semiconductor process representation (e.g., the Stanford FABLE system), the need to represent multiple abstraction levels or views for process information was recognized, leading to language descriptions that are somewhat different from those in conventional programming languages (e.g., through the definition of different code branches or blocks that are only applicable to fabrication interpretation or simulation interpretation). At the other extreme of the spectrum lies the knowledge or object-based analogy and approach, where the engineer locates and adapts (e.g., by specializing or overriding default attributes) existing process objects and wires these together to construct larger objects or flows.

The process flow language approach can be implemented in as simple a fashion as text files that are interpreted by different kinds of application programs to accomplish specific subsequent tasks (e.g., to run fabrication or simulation tools). On the other hand, the object-based approach immediately suggests a graphical user interface whereby the user manipulates some visual representation of process step and flow objects and specifies or edits their characteristics. In practice, the distinction between these approaches is not so sharp. Integrated programming or design environments also support the building of libraries of process steps and flows, as well as guide the graphical editing of either procedure parameters or object attributes. Additional search and design aids are typically desired in both approaches. Maintenance of consistency (e.g., between versions of steps and process flows that use those steps) is important; process design rule checkers that verify that safety or other requirements are satisfied are also often provided.

Another element—document management—has often been integrated with process representation systems. In many environments, the careful management of versions of process steps and flows is important, together with authorization or sign-off procedures.

Program Interfaces

The definition of standard program interfaces is of critical concern for the development and integration of the many application programs that need access to process information. At this stage, most contemporary process representations adopt an object-based approach. Each process component typically has multiple views that contain attributes (or name-value pairs) that can be accessed and manipulated. Clearly, one set of these attributes must capture the process sequence or hierarchy. Other views typically align with the generic process model or with other sets of attributes needed to accomplish specific tasks. Conventional object and object attribute creation, access, and mutation methods are typically provided.

Historically, many such programming interfaces have been language and implementation specific. Generic object models and standard interface description languages, however, enable the specification of language-neutral interfaces to common data stores. Indeed, approaches such as the Object Management Group (OMG) Common Object Request Broker Architecture (CORBA) also enable remote network access to such information from multiple applications implemented in widely different programming languages. At present, examples of such interfaces have been demonstrated, and the semiconductor industry is working to define standard interface definition language (IDL) specifications so that independently developed utilities and systems can interoperate.

PROCESS LIBRARIES

It can be argued that process knowledge capture alone is sufficient to motivate development of a process representation. Indeed, a huge concern is that substantial experience and knowledge about the process generated during design stages is lost and never transmitted to the manufacturing effort. A detailed process representation, particularly one that can incorporate both formal and human digestible information (e.g., scanning electron microscope [SEM] images of typical or problematic cross sections, simulation results), can serve as a shared repository for knowledge about the process.

The development and support of process libraries is an important part of virtually every process representation system developed to date. Part of this stems from the desire to unify both design and manufacturing views of the same information. Another motivation is the inherent complexity of semiconductor fabrication, which dictates that as much as possible of previous process steps and flows be reused when creating a new process. Finally, it is often the case that process design occurs with physical or logical separation between the unit or module process designer and device or integration engineers, who must assemble or adapt individual process steps to create an integrated flow that produces the desired devices. A more recent trend is the increase in use of foundries, which provide fabrication services; to support both conventional complementary metal oxide semiconductor (CMOS) and unconventional (e.g., microelectromechanical system, or MEMS) foundry capability, the need for network-accessible process repositories (including statistical, design rule, and simulation information) will likely increase in the future.

APPLICATION PROGRAMS

The full power of a process representation is only realized when a collection of computer programs or applications is available to accomplish the wide range of tasks that need process information. For example, various programs may produce reports or subsets of information from a full process repository or process flow for different purposes or targeted to specific types of users, such as process designers, managers, schedulers, equipment suppliers, control engineers, and others connected to a fabrication process. Among the most common of these report generators are programs that create a paper or electronic traveler or run sheet that provides summarized process sequence, equipment to run on, key process parameter information, and room for measurements for specific wafers or lots that travel through the fab.

When ICs are fabricated, the run sheet or traveler follows along with each batch or wafer lot and contains the sequence of processing steps need to produce the IC. In a computerized manufacturing system, the traveler may be a computer record rather than exist as a printed paper record that travels with the wafers. In either case, a simple implementation of a process traveler may be a report generated from the more formal process representation.

The run sheet is typically augmented with some of the key details that are unique to the machines being used or to the product being manufactured. These augmentations may include, for example, the recipe number to be used for a particular machine, summaries of expected results (e.g., the thickness of an oxide to be grown), and measurements to be taken. Data concerning the processing and the resulting wafer state are usually collected throughout the actual processing. At times these data are recorded on the run sheet itself, while in other cases the measurements are recorded in computer files or records, and the filename or data path is noted on the traveler. In a more sophisticated implementation, the electronic run sheet is itself an interactive computer application, both providing processing instructions from the process flow and accepting data input from operators or equipment.

APPLICATION: FACTORY DESIGN

In modern fabs, CIM systems fundamentally depend on process information (among a great deal of other information). Less well known, but important to note, is the need for process information well before a factory is constructed. Process flow information is also needed in factory design and planning (e.g., capacity planning to define equipment set requirements to meet production goals for various mixes of products and processes).

Answering the simple question of which machine types will be needed in a new facility requires knowledge of the process. Determination of the capacity of a factory involves knowledge of the product mix desired as well as detailed process knowledge. Interactions between sequential machine operations can be simulated or otherwise determined from the process, and material transport systems can be designed and tested based on process flow knowledge.

APPLICATION: COMPUTER INTEGRATED MANUFACTURING

The management of a modern IC fab is an enormously complex endeavor, and the representation of the process plays a critical role in the CIM system that accomplishes this task. Indeed, some CIM systems, such as the Computer-Aided Fabrication Environment (CAFE) system developed at MIT, have been constructed with the process representation as the key organizing or integrating mechanism. In other modern CIM systems (e.g., those provided by Consilium, Promis, Texas Instruments, and Fastech), a process representation interacts with the CIM system in three key areas: flow and resource scheduling, recipe execution, and data collection and analysis.

The CIM system must manage the flow of material and information in the facility. In addition to details about the physical process (as described in the generic process model), the process representation must also indicate those resources that are needed to accomplish each process step. Typically, these resources are specific equipment (or classes of equipment) that are to be used in the step, as well as mask, tooling, or materials required at that step. Other resources may also be indicated; the time required to complete the step is of clear importance. The CIM system (or modules within the CIM system) will schedule wafer movement, allocate and mobilize other resources, and communicate with material handling systems, equipment, and operators to execute the specified processing.

As part of the process execution, modern CIM systems will often download detailed operating specifications or programs on fabrication equipment. While a few research systems have integrated generic process step specification with individual equipment program download, at present most CIM systems and process representations act more as a holding place for these detailed recipes (e.g., by uploading a recipe developed directly on the tool). The more complete integration of detailed equipment recipe process representations remains a goal for the future.

In addition to process execution, the CIM system must gather measurement information for use in future decision making, debugging of the process, as well as quality management and continual improvement. Again, the process representation plays a central role; the process representation will typically indicate precisely what metrology is required and link that information with the corresponding process step and flow. In fully integrated systems, the diagnostic tools can seek out correlations between observations and process specifications both within and across multiple process flows.

APPLICATION: SIMULATION

A process representation must support not only active fabrication; it must also enable simulation of both facility operation and of the effects on the wafer for process/device engineering and yield optimization.

Operational Modeling

To model capacity, throughput, and other key operational aspects of an existing or contemplated IC fab, descriptions of the process are required. The key information needed is one subset of a complete process description and typically includes at a minimum the sequence or flow of process steps, with clear identification of the specific equipment (or classes of equipment) needed for each step, and the time required. More detailed information may also be useful, including time dependencies based on batching, setup, or other handling, and other resources required, including operators, handling equipment or tooling, and materials or consumables. Because a substantial volume of such process information may be required for operational modeling and facility management decision making, it is highly desirable that information be represented in a form that discrete event or operational simulation and modeling tools can use so that reentry of this information can be avoided.

Process Modeling

To model the physical processes at work in semiconductor fabrication, a great deal of detailed unit process information is typically required. Physical process simulators generally operate from the treatment view of a process step (that is, the description of the physical environment of the wafer during processing). However, a complete physical description of the environment adequate to predict process results may not be available. Moreover, even if such a description is available in principle, it may be outside the modeling capability of a particular simulator. Hence, more limited or ad hoc models, requiring significant empirical calibration, are often used in simulation.

The central historical difficulty has been the existence of a multitude of alternative and unique representations of the process. Each simulation tool has typically defined its own file format or user interface for expression of those details of the process that it needs. An early approach that sought to bridge those gaps was proposed by MacDonald et al., where simulator statements were inserted into comments fields in the process routes descriptions in a preexisting CIM system, enabling in-fab process engineers to perform simulations to assist in process diagnosis and improvement. In this approach, the process sequence or flow is shared between fabrication and simulation, but process details must be entered in both step recipes and simulation statements (that is, no other parameter coupling between the manufacturing and CAD information was achieved; only the correspondence of steps was achieved). Another approach was reported by Durbeck et al. in which an on-line specification management system primarily targeted at managing process libraries with engineering check off and subsequent generation of run sheets was extended to generate parameterized simulator input decks. In this case, process details are entered only once into the system; and these values automatically propagate to the views or descriptions needed for simulation. The approach remains limited, however, in that it is only possible to emit or generate files for use in conventional process simulators.

Several process representations have been focused on supporting process simulation environments. In these cases, a key goal is often to capture process sequence and key process parameter details needed for more than one process simulator. A difficult issue has always been the handling of simulator-specific details, such as model coefficients and gridding or timestep information. One approach is to encode such information directly into the process flow representation. For example, Wenstrand enabled multiple simulator views to be described in each process object, with key process parameters linked between these steps. An alternative approach is to suggest that such simulator-specific information is best separated from the intrinsic process information and instead should be treated by intelligent compilation or interpretation approaches. That is, an interpreter for some particular simulator would examine wafer state and process specification information and generate simulator information with appropriate gridding or model information (perhaps by consultation of additional simulator-specific library information). In either case, it is recognized that implementations must deal with existing simulators that are not ideal but rather require more than simple wafer state or treatment information.

Equipment Modeling and Synthesis

Clearly, an approach that fully integrates the information content shared between both fabrication and design requirements provides additional benefits and opportunities. A data representation with well-defined levels of abstraction enables multiple types of simulators and other CAD tools beyond those that simply produce wafer descriptions for later device simulation. Indeed, a picture emerges of the process representation as a dynamic representation where new knowledge about the process is stored as it is created by CAD tools or experiments. For example, equipment simulation may be able to take detailed equipment state or recipe information and predict what treatment or environment the wafer will see (and record that environment in the appropriate view of a process step). Another simulator might be able to summarize the effect of the treatment as a change in wafer state. In the reverse direction, synthesis tools or utilities could also generate more detailed descriptions from abstract specifications. For example, for a desired treatment, a recipe synthesis tool could generate the detailed recipe or settings needed to create a desired environment around a fabrication tool, given the constraints of existing equipment in a particular facility. The representation and capture of the full spectrum of process information between both fabrication and simulation enables improved process development in other ways as well. Direct comparison of simulation with historical measurement (perhaps including measurements or characterization data spanning original process development as well as manufacturing) can accelerate yield learning and diagnosis. While experimental systems have achieved substantial integration (e.g., the CAFE CIM system developed at MIT), support of such integrated capability is not yet present in commercial systems.

STANDARDIZATION

Various efforts have been made to establish industry-wide standards for process representation, but none have yet achieved widespread acceptance. The most promising approach to standardization at present seems to be through the definition of application programming interfaces (API) specifying how other programs (CIM systems, CAD tools, etc.) manipulate process information. In particular, the OMG CORBA specification enables platform and programming language interoperability, so that applications and services may be written in various languages and run on computers with different operating systems. This is the approach taken, for example, by the SEMATECH CIM framework. In addition, equipment communication and recipe management standards for upload and download of process details are under development. These promise to move the industry toward standard process flow representations and enable the development and deployment of a new generation of interoperable CIM and CAD systems.

BIBLIOGRAPHY

D. Akkus, *Process Advisors: Process Synthesis for Arbitrary Initial Conditions by Analytical Models,* Bachelor's thesis, Massachusetts Inst. Technol., Cambridge, MA, 1990.

D. S. Boning, *Semiconductor Process Design: Representations, Tools, and Methodologies,* Ph.D. thesis, Dept. Electr. Eng. Comput. Sci., Massachusetts Inst. Technol., Cambridge, MA, 1991.

D. S. Boning et al., A general semiconductor process modeling framework, *IEEE Trans. Semicond. Manuf.,* **5**: 266–280, 1992.

D. Durbeck, J.-H. Chern, and D. S. Boning, A system for semiconductor process specification, *IEEE Trans. Semicond. Manuf.,* **6**: 297–305, 1993.

C.-Y. Fu, N. H. Chang, and K.-K. Lin, Smart integrated circuit processing, *IEEE Trans. Semicond. Manuf.,* **2**: 151–158, 1989.

S. B. Gershwin, Hierarchical flow control: A framework for scheduling and planning discrete events in manufacturing systems, *Proc. IEEE,* **77**: 195–209, 1989.

C. J. Hegarty, L. A. Rowe, and C. B. Williams, The Berkeley Process Flow Language WIP System, Tech. Rep. UCB/ERL, M90/77, UC Berkeley, 1990.

C. P. Ho et al., VLSI process modeling—SUPREM-III, *IEEE Trans. Electron Devices,* **ED-30**: 1438–1452, 1983.

R. A. Hughes and J. D. Shott, The future of automation for high-volume wafer fabrication and ASIC manufacturing, *Proc. IEEE,* **74**: 1775–1793, 1986.

P. R. Kristoff and D. P. Nunn, The process specification system for MMST, *IEEE Trans. Semicond. Manuf.,* **8**: 262–271, 1995.

A. J. MacDonald et al., Integrating CAM and process simulation to enhance on-line analysis and control of IC fabrication, *IEEE Trans. Semicond. Manuf.,* **3**: 72–79, 1990.

M. B. McIlrath and D. S. Boning, Integrating process design and manufacture using a unified process flow representation, *Proc. 2nd Int. Conf. Comput. Integrated Manufacturing* (Troy, NY), Los Alamitos, CA: IEEE Computer Society Press, 224–230, May 1990.

M. B. McIlrath and D. S. Boning, Integrating semiconductor process design and manufacture using a unified process flow representation, *Proc. 2nd Int. Conf. CIM,* Troy, NY, 1990.

M. B. McIlrath et al., CAFE—The MIT computer-aided fabrication environment, *IEEE Trans. Compon. Hybrids Manuf. Technol.,* **15**: 353–360, 1992.

M. B. McIlrath et al., CAFE: The MIT computer-aided fabrication environment, *Proc. Int. Electron. Manuf. Technol. Symp.,* Washington, DC, 1990.

H. L. Ossher and B. K. Reid, FABLE: A programming language solution to IC process automation problems, *Proc. SIGPLAN 83 Symp. Programming Language Issues Softw. Syst.,* **18** (6): 137–148, 1983.

H. L. Ossher and B. K. Reid, *FABLE: A Programming Language Solution to IC Process Automation Problems,* Tech. Report 248, Comput. Syst. Lab., Stanford Univ., 1985.

H. L. Ossher and B. K. Reid, Manufacturing specification, *Proc. 2nd Annu. IC Assembly Autom. Conf.* (INTEM), 1986.

J. Y. Pan, J. M. Tenenbaum, and J. Glicksman, A framework for knowledge-based computer integrated manufacturing, *IEEE Trans. Semicond. Manuf.,* **2**: 33–46, 1989.

C. Pichler and S. Selberherr, Process flow representation within the VISTA framework, in S. Selberherr, H. Stippel, and E. Strasser (eds.), *Simulation of Semiconductor Devices and Processes,* vol. 5, Vienna: Springer-Verlag, 1993, pp. 25–28.

L. A. Rowe, C. B. Williams, and C. J. Hegarty, *The Design of the Berkeley Process-flow Language,* Tech. Rep. No. 90/62, Electron. Res. Lab., UC Berkeley, 1990.

P. Saha, *IC Process Synthesis by Analytical Models,* Bachelor's thesis, Massachusetts Inst. Technol., 1989.

S. D. Senturia et al., A computer-aided design system for microelectromechanical systems (MEMCAD), *IEEE J. Microelectromech. Syst.,* **1** (1): 3–13, 1992.

J. S. Wenstrand, *An Object-oriented Model for Specification, Simulation, and Design of Semiconductor Fabrication Processes,* Ph.D. thesis, Stanford Univ., Stanford, CA, Technical Report, ICL-91-003, 1991.

J. S. Wenstrand, H. Iwai, and R. W. Dutton, A manufacturing-oriented environment for synthesis of fabrication processes, *IEEE Int. Conf. CAD,* ICCAD-89, 376–379, 1989.

C. B. Williams, *Design and Implementation of the Berkeley Process-flow Language Interpreter,* M.S. thesis, UC Berkeley, 1988.

Donald E. Troxel
Duane S. Boning
Michael B. McIlrath
Massachusetts Institute of Technology

SEMICONDUCTOR RADIATION DAMAGE. See Radiation effects.

SEMICONDUCTORS. See Electronic components.