What Can SFF CAD Learn from the VLSI CAD Revolution?

Carlo H. Séquin and Sara McMains
CS Division, U.C. Berkeley

1 VLSI Design and Fabrication

Before forging ahead with new standards for SFF, it seems worthwhile to look at design and fabrication in the domain of integrated circuits and to review the VLSI CAD revolution of the 1980s. There may well be some valuable lessons that carry forward to the domain of SFF.

IC fabrication is a 2.5-dimensional technology. There are many different classes of processes: NMOS, CMOS, Bipolar, BiCMOS, GaAs ... In each class there are many variations among different fabrication lines. But in all cases the principle is the same: one starts with a flat wafer, usually a silicon crystal lattice with a single orientation, and then produces a series of changes in areas that are defined by the geometrical patterns on a sequence of fabrication masks. In order to produce a "Si-gate NMOS" circuit (a vertical layering of Metal and Oxide, on top of a Semiconductor which has regions of Negatively charged carriers), one might use the following key conceptual steps. One mask pattern (often symbolically shown in green color) and the associated processing steps may cut open a pattern of trenches into the oxidized silicon crystal surface, into which small amounts of "impurities" are diffused in order to produce regions that are more conductive than the bulk of the silicon substrate. A second (red) pattern may define connection strips in a deposited polycrystalline silicon (Poly-Si) layer on top of the previous structure. Where these strips cross the "green" trenches, transistors are formed: the voltage on the red strips – or "Si-gates" – controls the amount of conductivity in the green trenches underneath. A fourth (blue) pattern defines another layer of interconnections, in a metal layer on top of the whole processed wafer. A third (black) pattern specifies where connection holes should be cut through the insulating layers placed between the "red" and "blue" interconnection levels.

For a given class of processes, the basic functionality of the various layers is always the same, even though the thicknesses of the many layers and the exact cross-sectional shapes of the trenches and conductor strips may vary. In a different process class, the levels may perform different functions; for instance, in an older "metal-gate NMOS" process, the amount of conductivity in the semiconductor bulk layer is controlled by gates formed in the metal layer. Whoever designs the masking patterns for a particular circuit has to understand what the basic functions are of the various patterned layers. In the early 1970s the design of a circuit layout was an "art" where each designer had to work closely with the people operating the fabrication line in order to define appropriate patterns that together with the envisioned processing sequence would then produce the desired circuit behavior.

One of the contributions of Mead and Conway ["Introduction to VLSI Systems," Addison Wesley, 1980] was to push for a simplification and standardization of this interface between designers and fabricators. For many applications, the performance of the digital logic circuits is not very critical compared to the capabilities of the technology and will be good enough even when the potential of the chosen fabrication process is not used to its limits. It was thus possible to define a simple set of geometrical layout rules that would assure that functional transistors and circuits of acceptable performance would be produced for fabrication lines whose processing parameters were kept within reasonable parametric ranges. This was done first for the Si-gate NMOS process and later for "CMOS" technology (Complimentary MOS, with transistors of both polarities). This simplification allowed an abstraction of the design process, where geometrical "recipes" would produce transistors of desired performance (speed, driving power) or circuits of desired functionality (AND-gates, NOR-gates, multiplexors ...).

To make this abstraction work, it is crucial that the semantics of the various patterns is well defined. After some initial controversy, the geometrical shapes were defined to represent "what you can see when you look down on the final IC chip." Since the actual masks used in fabrication may actually have different dimensions, owing to expansion or shrinking of the various features during the fabrication process, or owing to size changes when masks are copied, it became the responsibility of the fabrication houses to compensate for all these cumulative effects and to produce the masks for their own use that would deliver the specified geometrical dimensions on the final chip. Thus the fab-line dependent variations and idiosyncrasies became invisible to the circuit designer, and the same set of geometrical pattern specifications became portable from one fab-line to another. As technology improved and it became feasible to make ever smaller features and thus faster and more compact circuits, it was even possible to take the old patterns and to shrink them automatically to make them usable again for these advanced processes. At this point one has reached a true abstraction of the layout design process, where the designers may not even know what the dimensions are of the rectangles that they are drawing.

2 Higher Levels of IC Abstractions

After the lowest level of abstraction of implementing pieces of integrated circuits (by specifying the raw geometry of all the features on the chip) became routine, and ever larger collections of useful and reliable circuit elements had been compiled, it became practical to shift the level at which designers would think about their circuits to a higher level. The logic level had already become well established with the library of bipolar circuits described in Texas Instruments' famous TTL Data Book. The semantics or the functions of the various gates was quite clear, and library cells that implemented these various useful logic circuits could readily be designed and optimized for different fabrication processes. The designers now simply had to plug together these "logic gates" into more complicated digital subsystems.

Very soon, frequently used higher-level circuit components emerged: registers, datapaths, memory blocks, arithmetic-logic-units (ALUs); these became the natural – and often parameterized – building blocks for larger systems such as processor architectures. This "register-transfer-level" of abstraction is most conveniently used when a designer tries to trade off system speed against implementation costs and tries to find the most cost-effective degree of parallelism in the architecture.

Between the layout level and the logic level, there are a couple of other abstraction levels that were often used by many designers. First there is the "sticks" level in which the connections in the various layers of the integrated circuit are not represented by their exact geometry, but only by their topological ordering and arrangement on the chip surface. This level can be drawn more quickly and frees the designer from worrying about exact dimensions and about layout-rule correctness. A computer program, called a "compactor," then converts a sticks diagram into a densely packed, layout-rule correct geometric layout. However, this level never became useful for exchanging designs between designers in different groups, since the semantics of this description was never clearly defined and standardized.

Similarly, yet another abstraction level, the "switch" level, never played a mayor role as an interchange standard, even though this level is very important for circuit simulations and for optimizing the performance of a generic circuit. Too many parameters would have to be specified with each component at the switch level, and the actual performance would then still be dominated by many parasitic circuit elements, such as stray capacitances that are strongly affected by the exact layout and by the relative arrangements of the circuit components on the chip.

Above the register transfer level, there are also further levels of abstraction. But as one climbs higher up in this abstraction hierarchy, the concerns of the designers become richer and harder to define. At the "functional" or at the "behavioral" levels, one tries to formally capture what the system overall is supposed to do – but this is rather difficult. For a long time, certain semantic issues had been left unanswered: for instance: “What is the meaning of time in this domain ? Is it continuous or discrete ? Is it fully synchronous to some clock ? And if so, how are external asynchronous events handled ?” The lack of semantic clarity and completeness made it difficult to exchange systems specifications at this level. In the last few years, however, a standard language (VHDL) with reasonably well defined semantics has emerged.

3 Conversion and Checking Tools

Defining these languages and interchange formats with proper semantics is just the beginning; they are not much use unless we also have convenient tools to create descriptions at these various levels and to convert these descriptions from one level to another.

Programs that convert descriptions from higher levels of abstraction to lower, more detailed description levels are known as “generators” or “compilers.” For example, we might have a parameterized “N-bit full adder” generator, which – upon specifying a value for N – will generate an adder circuit with N bits in a particular technology. This relieves the individual designer of the tedium of explicitly specifying all the geometric details of the layout of such a circuit.

The inverse process, finding a higher-level description from a large – possibly unstructured – mass of lower-level primitives is known as “extraction.” This is often used as a verification step after a lengthy, multi-stage compilation process, e.g., to verify that a large collection of layout rectangles indeed constitutes a circuit with the desired properties. Such a circuit-extraction process may also find spurious circuit elements that never show up explicitly in the generation process. For instance, long interconnection lines may have associated with them a significant amount of capacitance to the substrate. To adequately model the performance of such a “transmission line,” this capacitance has to be calculated and suitably incorporated into the model of the circuit.

To close the verification process, the extracted circuit would have to be compared to the original, intended circuit. This can basically be done with some graph-matching algorithm; however, some margin of error has to be tolerated in the parameter values for each individual circuit component because of the parasitic geometries mentioned above. There is another difficulty: the same flat circuit or logic description can often be hierarchically grouped in different ways that are equally valid or “natural.” The extracted hierarchy may then be quite different from the one originally conceived by the designer, and thus hard to compare. It might thus be worthwhile to leave “hints” or other informal information in lower level descriptions that convey the original design intent.

There are also internal consistency checks that can be made on individual descriptions alone. For instance, in a circuit diagram, one can check that differently labeled signal or power lines are not short-circuited to one another, or that all terminals with the same labels are indeed internally connected. At the level of the layout geometry, one would typically apply some geometric checks to see that the chip can be fabricated with reasonable tolerances and will still yield – with high probability – the geometric patterns that the designer had in mind. Primarily, such layout checks would test for certain minimal separations between elements that should not fuse together, and for certain minimal internal dimensions for paths that should not be broken. Comparing the geometry on two different levels, one would check for sufficient tolerances in the mutual overlap of features, so that one retains coverage even in the presence of small mask misalignments. All these checks are tightly coupled to the semantics of the features represented and to an understanding of the capabilities of the processes that are controlled by these descriptions.

4 What Can We Learn from This ?

Many similarities become apparent when looking at some of the SFF processes in the context of the above experiences with VLSI. Here we present our initial reactions to the questions raised in the call for participation for this workshop. In many instances where the situation in SFF is sufficiently different from that in the domain of VLSI, we are not ready to take a definite stand; we then simply present the pros and cons of an issue to help focus the discussion at the workshop.

4.1 Modeling and Design Exchange

> *What level of abstraction should be used for describing the physical design? Should it be two-dimensional layers corresponding to the layers which are built during the fabrication process? Or should it be a form of a three-dimensional description with features identified?*

A single level of abstraction will definitely not be sufficient. There is a need to describe mechanical systems and their physical parts at one or more high-level abstractions that capture various aspects of their functionality. These higher levels of abstraction are outside the concern of this workshop.

SIF:
However, at some point, the shape of an individual part has to be defined, and a clean abstraction level is needed to describe the geometry of this part in a completely fabrication-process-independent way. This geometric shape specification should clearly be a 3D description. Many existing solid modeling languages would be quite adequate for this purpose. They should have the power and expressibility to describe smooth surfaces in a resolution-independent, non-tessellated, compact form, and should be able to represent cylinders and spheres perfectly. NURBS seem to be one of the obvious choices for that purpose.

However, the variety of existing solid modeling systems and the possible complexity of the constructs they offer may be bewildering and scary to the manufacturing people who should be able to accept and process models in this form. It might thus be worthwhile to define a lean and clean subset language that contains only the really essential elements and has unambiguous semantics. For discussion purposes we will call such a language "SIF," or "Solid Interchange Format." This should be a compact, human-readable ASCII language in the spirit of CIF. The format should be hierarchical to avoid the waste of transmitting repetitive data.

"Features" (e.g., a "hinge," or a "parameterized dove-tail slot") – like functionality – do not belong at the SIF level, but at a higher symbolic level. While including such features may make the job of the process planner and the rule checker easier (see Weiss and Prinz draft position paper), it may be too daunting to define a rich enough catalog of features for all current and future application domains. (Possibly, if a high-level description exists that includes features, this information might be passed down to lower level descriptions to make the extraction of fabrication features easier. But we would then need to discuss to what degree such "hints" are an informal local help and to what degree they should be officially supported in SIF.)

In general, any such language should be kept as simple as possible, but general enough to do all the jobs envisioned, and extensible in case new needs emerge that cannot be handled by the original format.

L-SIF:
There is also a need for lower level part descriptions that are much closer to actual fabrication plans. For SFF, a natural candidate would be some 2.5D layered description that is comparable to the CIF description of mask levels for a particular fabrication process. For the sake fo discussion, we call this format "L-SIF," for layered-SIF. The L-SIF descriptions for one and the same part may be quite different for different SFF fabrication processes.

Ideally, L-SIF would have a very similar style and semantics as the SIF language and should in principle be a derivative thereof. On the one hand, it might contain fewer geometrical constructs than SIF, since it might only have to express the 2D shapes of subsequent layers; thus, rather than having to describe NURBS surfaces, it would only have to express NURBS contours. On the other hand, the L-SIF languages for different SFF fabrication classes may need to convey some extra information such as the direction of a material-depositing nozzle movement. For some processes, such as Shape Deposition Manufacturing, the L-SIF process plan description might require almost the same richness of geometrical information as the idealized SIF part specification, since the individual layers are so thick and individually machined around their perimeters that they must be considered 3D solids in their own right.

Finally, below the level of L-SIF, there will be a plethora of machine-dependent control languages over which our task force will have little influence.

> *What type of model should be used to represent designs? What should be the role of traditional solid modeling? What attributes should the model provide in describing the design other than geometry? Candidates include strength, material, tolerance, etc.*

Traditional solid modeling systems have mostly relied on polyhedral or on smooth "boundary representations" (B-REP) or on "constructive solid geometry" (CSG) assemblies of a few clean primitives (cubes, cylinders, spheres...), or of a combination thereof. While other models exist for describing geometry that are mathematically purer, it is not clear that they are preferable to describe shapes at the SIF or at the L-SIF level or that they would be accepted by a generation of engineers that grew up with today's CAD and modeling tools.

For cleanliness and compactness of the description it may well be desirable to allow SIF descriptions to be a combination of B-REP and CSG: i.e., regularized boolean set operations trees whose leaves are B-REPs of 2-manifolds, half spaces, and partially bounded objects. This would allow, for instance, a compact, resolution-independent specification of a shape defined by one or more general NURBS with some cylindrical holes in it.

At the SIF level, the different geometrical regions defined would have to carry information on the "desired" volume or surface properties, which the manufacturing processes would then have to approximate as closely as possible. Material selection, density, color, tolerances, and surface roughness may be such properties that need to be specified. This could be accomplished by using a volume statement that takes a solid region and specifies the "content" or materials properties of that domain. Here is an example that demonstrates the use of CSG and of several geometric regions occupying the same space: graded coloring could be specified in a large bounding box that contains the whole part and which is formally intersected with it.

At the L-SIF levels, the volume or boundary information may then be much more dependent on the particular SFF process envisioned and specifically refer to some of the available materials that could be dispensed in each layer, or to the particular local treatment that these layers may experience at certain locations.

Specifying the material at the SIF level raises some interesting issues. Since different processes have different material capabilities, a design that includes a material specification is no longer process independent. For a manufacturing process where SFF will only be the first stage of the tooling, such as making wax positives for investment casting, the material the SFF uses won't be the material of the final part. But for a different SFF process that can make parts directly out of metal, the SFF will use metal instead of wax to produce what will ultimately be the same part. (We should probably limit ourselves to specifying the properties and materials of the primary part emerging directly from the SFF process.)

A key issue is that of auxiliary support structures that are not part of the final geometry but are required for a specific manufacturing process. For example, wax patterns for investment casting need to be attached to the sprue, gates, and runners in order to distribute the molten metal, while wax patterns that are to serve as conceptual models would not need this additional geometry defined. Similarly, a part produced by stereolithography (SLA) will need support structures for cantilevered portions of the model, whereas these would be unnecessary if the part were produced by selective laser sintering (SLS). Generating such "supporting" geometrical features is not yet fully automated. While the designer may have to think about the role that these supports play in her part design and how the supports will get removed and how the break-off ridges may get smoothed, the actual placement of the support geometry is normally done by the experts at the fabrication service, since they are much more knowledgeable about the needs for supports in their particular process than the designer.

Another issue that may get addressed quite differently at the SIF level and in the L-SIF descriptions concerns the way one may specify embedded components. At the SIF level, the designer may simply specify the location of the component and it could then be inferred that the embedding material has a corresponding cavity at that location; alternatively, a suitable cavity could be specified explicitly with a boolean difference operation. At the L-SIF level, the cross section of such a cavity may appear explicitly on each layer; alternatively, an explicit pause statement at the layer touching the minimum z-location of the part to be embedded could be generated by the slicer program that creates the L-SIF description from the higher level SIF part description. The .STL format currently has no explicit way to handle such embeddings, making it necessary for the designer to convey her intent to the fabrication house separately. The SLA machine operator then adds a "pause" statement to the control code at the layer where the embedded component will be placed, and then manually inserts the desired component.

What should be the form of a design exchange format? Can the format support alternative SFF processes? Is there a common set of information required by all SFF processes?

Ideally, the exchange format should be at as high a level as possible and should be process and machine independent; this is the level of SIF which specifies **"what"** shape is desired, but ignores **"how"** this is accomplished. However, this approach is appropriate only if the worst case process inaccuracies are smaller than the finest feature that the designer is concerned about.

As the design specifications get closer to the limitations of the technology, the idiosyncrasies of the fabrication process will have to be taken into account. Today, most SFF processes have quite different capabilities, and most are not even isotropic, so that the part orientation during the fabrication process becomes a crucial issue. A particular SFF process will thus have to be targeted and parameters such as part orientation, layer thickness, and auxiliary process steps such as curing or shot peening may have to be specified explicitly. Under those circumstances, it may be more appropriate to send a part specification to the fab-house in the form of a process plan at the L-SIF level. Suitable CAD tools must be developed to assist in this conversion process. The key is that each process and the corresponding conversion must try to approximate the desired geometry as closely as possible within the technological limitations.

Internally to the fab house, this L-SIF description may then be converted further into a sequence of instructions to drive a particular machine – again trying to realize the specified 2.5D geometry as closely as possible. This latter step should definitely be hidden from the client.

4.2 Design Rules and Tools

What type of design rules can be defined that when applied to the representation will guarantee successful fabrication in a series of SFF fabrication processes?

At the SIF level, we may employ generic tests to check that the representation describes closed solid objects (as already exist for .STL files). An other test would be looking for intersections of solids having incompatible volume property specifications, e.g., different materials.

At the L-SIF level, at the latest, different design rules must be applied for each different class of SFF process – otherwise fabrication will be limited to the least common denominator of process capabilities. Rules could include the minimum separation of surfaces that will be guaranteed not to fuse (e.g., to make a tight bearing), and the minimum wall thickness for a given geometry that will not punch through or collapse (e.g., to make a wine glass); this minimum thickness may be different for different orientations.

For SLA in particular, design rule checkers should test whether all parts of an object are anchored to the build platform during all stages of the build process, and whether features have adequate supports. Any process subject to curl could have rule checkers to test whether the curl distortion of the part is predicted to lie within acceptable limits.

> *What should be the nature of design tools to support SFF? VLSI research produced design capture, design rule checking, mapping of logical constructs into physical transistors, physical placement of transistors, and wire routing. Are there analogous tools for SFF including design capture, design critics to identify non-manufacturable features, and fabrication process planning?*

Clearly we want as much automated generation and compilation as possible to speed up the overall prototyping process and enhance its reliability. We also need extraction tools and certain verification tools at all levels. Like the design rules themselves, some design tools will be process specific. Process planners in particular have very different requirements for the different SFF technologies, and very different process planners are shipped with the various commercial SFF systems.

Another question that needs to be addressed is at what point in the process should the orientation for the final build be chosen? Different SFF processes may have different properties along different axes of manufacturing. Definitely the axis perpendicular to any layering is special, but even the remaining two axes may have different properties in some manufacturing processes. Designers may or may not know about these differences. If they do *not* know, they will have to design for a worst case orientation making the weakest assumptions for all directions. If they are very knowledgeable about the process to be used and its performance characteristics in different directions, then they will want to specify the exact orientation in which the designed geometry should be built in the process framework. A useful area of research might be the development of process specific tools that attempt to find the "best" orientation for a part, if the designer doesn't know. For processes that can build multiple parts in a single build cycle, the "best" orientation may also depend on the spaces left by the other parts being built in the same build cycle. Whether specified by the designer or derived automatically, design rules, simulation, and process planning will all need to know the orientation.

5 Conclusions

In the domain of VLSI CAD tools, the vision has always been "top-down" with a focus on the "holy grail" of a fully automatic "silicon compiler." However, the suite of tools that has actually made a difference and that has revolutionized the world has been built "bottom-up," with a first focus on simple plotting, circuit extraction, and layout-rule checking tools.

This was followed by synthesis tools that could automatically generate some lower description from the next higher one, and several of which could eventually be cascaded in a sequence of compilation steps.

The final round then was to include optimization in this compilation step; it was no longer good enough to just produce an acceptable solution – one wanted to obtain the "best" possible solution at the lowest possible "price."

We are convinced that this same evolution paradigm will also apply in the domain of SFF fabrication.