Design Objectives Document

OK Technical Committee

Nick English, OK
Susan Estrada, Qualcom
Paul Koch, Cadence
David Lan, TSMC
Ralph Lanham, Artisan
Wolfgang Roethig, NEC
Mahmoud Shahram, Synopsys

October 2003

Revision history:

Draft 1:	**July 31, 2003**
Draft 2:	**August 6, 2003**
Draft 3:	**August 20, 2003**
Draft 4:	**August 25, 2003**
Draft 5:	**September 2, 2003**
Draft 6:	**September 4, 2003**
Draft 7:	**September 9, 2003**
Draft 8:	**September 11, 2003**

Editor: Wolfgang Roethig

Open Kit Initiative Executive Summary

The Open Kit initiative is a collaborative response to the need for standards in the design kits that link custom IC design to fabrication processes.

The inefficiencies inherent in the custom IC design process are due largely to the fact that design tools require detailed process data. Design data historically has been tool specific, leading to prolific development of models to met the needs of each specific tool, without regard to other tools' requirements.

With the emergence of the standalone foundry industry -- and the multi-vendor design tool and IP chain -- design kits have become the essential link between the semiconductor manufacturer and the design teams that are developing IC-based products.

To date, standards are largely non-existent in the area of design kits. Open Kit will change this by standardizing the following: nomenclature, use models, interfaces, quality thresholds, and delivery structures and mechanisms. This will reduce the proliferation of incompatible solutions, wasted effort, and confusion currently rampant in the industry.

As Open Kit standards for content and interfaces are accepted, custom design costs will decrease and the ability to communicate essential functionality, performance, and predictability of processes will be simplified. Foundries will be able to offer fewer and more robust kits to expedite the adoption of new technologies.

Library and IP suppliers will reduce costs, work more interactively with the foundries, and deliver a higher quality set of products to their customers.

EDA companies will be able to write tools to specified data definitions and interfaces, thereby reducing low-leverage, tool specific efforts.

And IC designers will get working silicon faster, now being able to more quickly migrate designs and reuse IP.

Open Kit is an important first step to developing standards for the fundamental element of all custom IC design – the design kit – and establishing a basis for the efficient custom IC design chain in the future.

Background

IC design processes are notoriously inefficient due in large part to the bottom-up nature of the creation of the data and formats that are necessary for the design tools to do their job. Design data historically is tool specific because of the disjoint nature of independent tool developers inventing their own data models that met the needs of their tool but invented in large part independently of any other tool developers. In recent years, standards efforts have been mounted in other areas of design with mixed success but with forward progress.

Standards activities are mostly working where applied; but in the area of design data they are largely non-existent. Today, the elements that make up this design data are incomplete, unreliable and sometimes non-existent.

Throughout the evolution of the EDA industry, companies that purchased tools from commercial providers have had to install the data elements that represented the IC process that was the target of the intended design. Over the years that data became known as design kits, as a differentiation from the tools that made up the design flow.

These kits were initially an internal concern within Integrated Device Manufacturers (IDMs), with formalization initially taking place within those companies who supported custom designs by selected customers.

With the emergence of the standalone foundry industry and the formation of a disaggregate design and IP chain, design kits have become an essential link between the semiconductor manufacturer and the design teams developing products. To meet this rapidly developing need, design kits have been created by a variety of sources: foundries, EDA suppliers, design teams, and 3rd parties. These kits have been known by a wide variety of names and acronyms, with the most common and generally accepted being PDK (standing for process design kit).

However, while PDK may be a commonly used and accepted term for these kits, there has been considerable variety in what is included in a kit, and a lack of standards for representing whatever information they do contain.

Objective

The Open Kit initiative will put a strong foundation to the IC design process by beginning to clean up the incredibly messy data sets of the fundamental tools that make up the closest connection between design processes and silicon processes.

The mysteries surrounding the structure and uses of these data sets (e.g. files, tech files, models, numbers of unknown origin, etc., cause redundant work by multiple companies to create a reliable design flow to eventually produce working silicon. This standardization effort, if even moderately successful, will remove inefficiencies and nuisances within companies and, hopefully, will work in areas of minimal competition between vendors as well.

No one is happy with the state of the non-organization of this fundamental design data and there is no unifying science to guide the organization effort. Nevertheless, we must start somewhere. This is a continuation of standardization efforts that even though have had mixed results, are indeed helping the industry improve its processes

This proposal is the continuation of the effort by the industry to standardize in areas that offer minimal competitive value add; but are extreme nuisances to every participant in the IC creation process. The goal is to reduce the redundant work to support industry leading design flows with an order of magnitude less effort to build and maintain the PDKs that are the foundation of any design flow and methodology.

Benefits

By adopting a set of standard definitions for PDK content and interfaces, all companies will reduce their costs and increase their ability to communicate the essential functionality, performance, and predictability of their offerings to the rest of the industry. Foundries will be able to offer fewer, more robust kits and offer more complete kits during their new technology release process. Library suppliers will reduce costs and appreciate more lucrative licensing and royalty deals. EDA companies will be able to write tools to specific data definitions and interfaces. End users will have fewer variables to deal with to get working silicon and find a lot of the barriers to migrating designs and reusing IP to be removed.

Product Development Teams

Product development teams will realize 3 primary benefits from an increased level of PDK standardization. First, the completeness and consistency of process technology information and representation will mean fewer mistakes in getting silicon out. Standards enable a greater familiarity with the design data and much more control over the management and versioning of the design data releases and structures.

Second, the DA teams supporting product development activities will experience less confusion in comparing tools and have much more flexibility in choosing tools as well as being able to much more quickly ramp-up new tool flows. Every company in the industry has between three to ten engineers supporting design kit integration. This standardization effort will make them much more effective in that integration and support function.

Finally, VPs of Engineering will be able to make more effective economic decisions because there will be much less confusion in assessing the capabilities of various process technology and sourcing options. Because the understanding the true advantages and disadvantages of process choices, there will be much less risk in planning and executing new projects.

Foundries & IDM Wafer Fabs

Foundries and the wafer fab operations within IDMs will also realize benefits from increased PDK standardization. First, the clarity of technical communication using standard terminology and representations will improve the effectiveness of design teams. This will reduce the cycle time for completing designs and successfully moving them to volume production.

Second, standard technology representation will reduce the cost of supporting multiple tool vendors. There should be faster bring-up of new tool flows if EDA tool writers are writing to a standard data structure and set of parameter

nomenclature and anticipated default values. It should also be easier to support new tool flows as engineers become familiar with the standard set of base level information. Something as simple as knowing where information is located means a great deal in productivity and customer satisfaction.

A third benefit of increased PDK standardization will result from the configuration management among version of technology information, EDA tool, and PDK implementation. This has the potential of enabling better technology life cycle management from early access during technology development through design modifications of tried-and-true designs.

A fourth benefit is the ability to clearly communicate technology capabilities and advantages to design teams. With the low level noise removed from the ability to understand foundry design kits, the foundry's special offerings in areas that they may excel would be much easier to communicate to current and prospective customers with the potential for higher value revenue with less costly support.

Library developers

With increased PDK standardization there will be fewer different elements of the kits from different foundries to understand and maintain. Since many things in the kits will be common, there will be a much quicker understanding of process differences.

Because the structure and organization of the delivered kits will be standardized, knowing where elements are and what to expect from them will ease the insertion of kits into design flows. Common device level foundations will simplify simulation changes and ease regression testing of migrated library components.

EDA tool providers

The standardization of process technology information within PDKs will provide at least three benefits to EDA tool providers and probably many more. First, standardization will speed tool adoption, usage, and proliferation by reducing the overhead required to link tools to process technology information. This will also create opportunities for continued innovations in custom and analog design tools.

Second, standard PDKs reduce the cost of performing tool and flow regression testing. It will be less necessary to conduct testing across PDK dialects or have to manage through incomplete technology representations.

Finally, EDA companies will also have the opportunity to increase value with total solutions by combining the delivery of tools and PDKs for instant customer ramp up on specific tool flows. Additionally, PDKs could establish foundries as joint marketing partners to increase custom customer owned tooling (COT) business.

Simplification of Critical Tasks

The following list of tasks is intended to be faster and less expensive to accomplish once the proposed standard is recognized and accepted by key practitioners.

1. Expedite PDK creation/generation/validation
2. Facilitate PDK support/maintenance
3. Integrating blocks from multiple design sources
4. Migrating designs and libraries between technologies, including layout
5. Substituting a tool in the flow for a new tool with same functionality
6. Introducing a new tool functionality into a design flow
7. Comparing features & capabilities of different process technology choices

Technical Approach

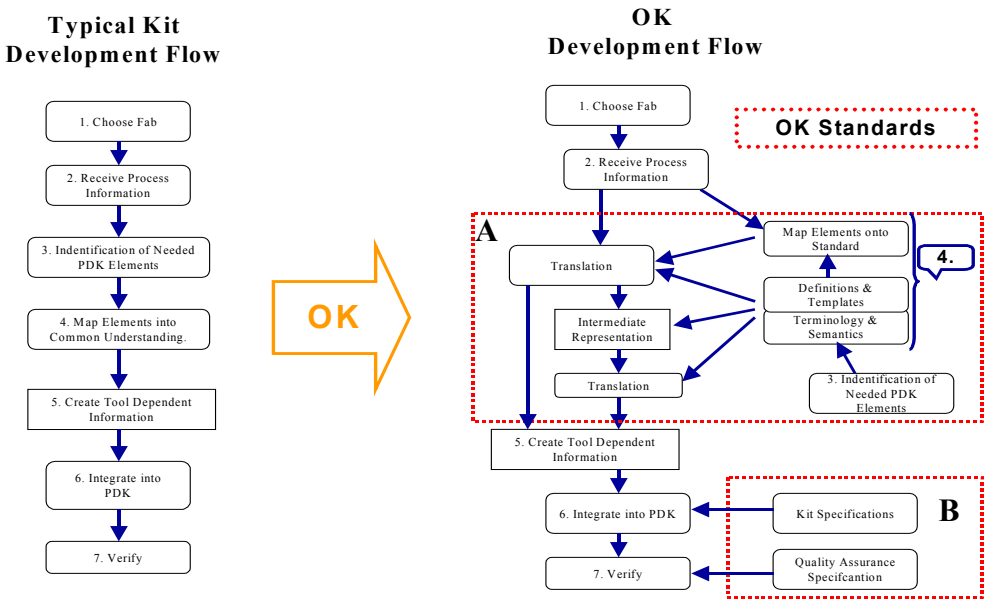
Fundamental idea

A representative flow for a design kit development effort today typically involves the following steps:

1) Selection of Process
2) Receipt of process information
3) Identification of needed Elements for PDK.
4) Review information needed for identified Elements.
5) Create PDK information typically in a Tool Dependent Fasion
6) Integrate PDK release
7) Verify

This flow is shown on the left side of Figure 1.

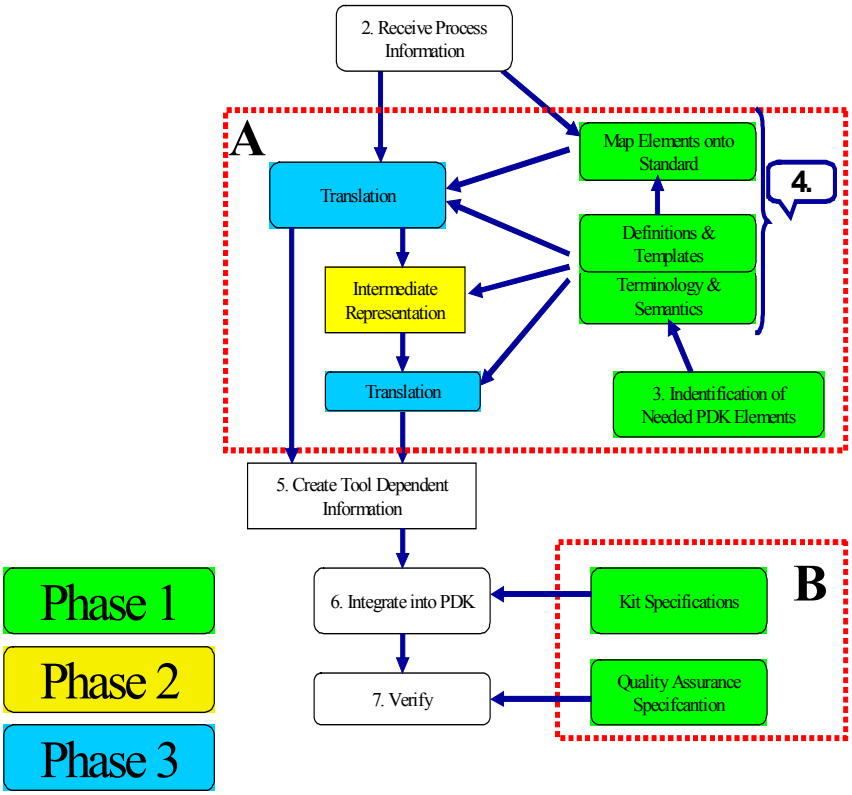
Figure 1 Towards an Open Kit development flow



In order to create a Standard Kit is desirable to identify the steps in the Kit development flow described in Figure 1 that can be abstracted and separated from where tool dependence is added to the process information.
In the right hand side of Figure 1 the dotted line box shows the area within a kit development flow subject to standardization. The dotted box “A” contains the steps in the process that define the elements in a kit and how the process

information will be captured in the standard. Giving standard definitions, meaning and standard translation mechanisms to be used in creating a kit. The dotted box “B” contains the definitions how the elements are combined and represented in a kit along with standards for quality assurance.
In order to gain traction with the standards effort the task described in boxes “A” and “B” must be done in a sequential order to insure focus and progress toward the goals. A phased approach is presented in Figure 2.

Figure 2 OK flow development roadmap



2

The PDK items subject to standardization can be classified in a multidimensional space. Within each dimension, the PDK items can be classified using discrete categories. The following list proposes four dimensions: task, tool, data, and data source. The intended items in the following list are categories associated with a dimension.

Task

- **Packaging**
- **Delivery**
- **Management**

Tool

- **XML as Information model vehicle**

Wizard of one form or another
Directory Structure

Data

Nomenclature
Files
Documentation

Data source

CDK supplier
Tool vendor

The categories within a particular dimension are mutually exclusive, but the dimensions themselves are orthogonal.

PDK definitions and terms

GENERAL TERMINOLOGY	
Term	Meaning / Context
custom circuit design	electrical design, physical implementation, simulation and physical verification of a circuit in which the basic unit of design are individual devices.
device	the basic design element supported by a semiconductor process technology, e.g. transistors, resistors, capacitors, inductors
structure	a building block that is the results from specific semiconductor manufacturing steps, e.g. implants, diffusions, contacts, interconnects, vias, that is used to construct a device. It is possible for structures to also be considered as device.
supported device list	the set of devices that are supported and controlled as part of a process technology definition; supported devices may be intentional or extracted.

DEVICE TERMINOLOGY	
Term	Meaning / Context
intentional device	devices which appear in the schematic diagram; these are the design elements of the circuit design
extracted device	devices which do not appear in the schematic diagram but which are added to the electrical circuit representation (netlist) through analysis of the physical circuit implementation.
main device element	the portion of a device (intentional or extracted) that performs its primary desired function or most significant behavior
parasitic device element	the portion of a device (intentional or extracted) that is present as a by-product of the physical implementation of the main device elements
principal device(s)	the 1 or 2 devices in a technology definition around which the architecture of the process technology is defined, designed, and optimized e.g. NMOS and PMOS devices in a digital CMOS process. Principal devices have the tightest specification tolerances.
primary device(s)	devices in the supported device list that is built from the structures defined for principal devices with specific additional structures, usually involving one or more dedicated, controllable process steps, e.g. poly resistor built from gate poly with dedicated silicide block mask and implant
secondary device	A device in the supported device list that is built from structures that are defined, designed, and controlled for principal and/or primary devices; these are sometime referred to as “free devices” or “intentional parasitic devices”, e.g. using metal interconnect to create an inductor or capacitor. Secondary devices typically have the widest specification tolerances.

Term	Meaning / Context
device class (category)	general functional category to which a device may be assigned, e.g. MOS, bipolar, resistor, capacitor, inductor, etc.
device type (vertical)	the unique combination of vertical structures that are used to construct and differentiate devices within a class, e.g. poly1 resistor, poly 2 resistor
device style (lateral)	lateral variations within a device type that have the purpose of achieving specified differences in device performance, e.g. modifying lateral layout of EBC of a bipolar transistor to achieve different Va specifications
device size	special case of style in which the differences in behavior are considered scalable, e.g. modifying gate width in a MOS device

MODELING TERMINOLOGY	
Term	Meaning / Context
model	the representation of a device used by simulation to represent its function and performance characteristics; models are made from a combination of general behaviors and parameters
model general behavior	portion of a model that represents the generic behavior of a device class and type; can be made from compact model equations, behavioral models, and/or sub-circuits
model process parameters	portion of a model that makes its general behavior and performance related to a specific process technology; e.g. model cards.
instance parameters (allowed design variables)	the properties assigned to an instance of a device that are passed to the simulator and model through the netlist and which describe directly controlled items like device size (e.g. L & W for a MOS device) and calculated items like sidewall capacitance.

EDA RELATED TERMINOLOGY	
Term	Meaning / Context
primitive	the smallest constructor element that is represented within an EDA system; primitives are used to build up device representations
instance	a uniquely identifiable placement of a device within a design
property	information that is associated with and assigned to primitives that specify or modify their behavior
instance property (direct, indirect)	a user-controlled (direct or indirect) parameter associated with an instance; direct properties are assigned by the designer, indirect properties may be calculated or derived from direct properties; sometimes called “designables”

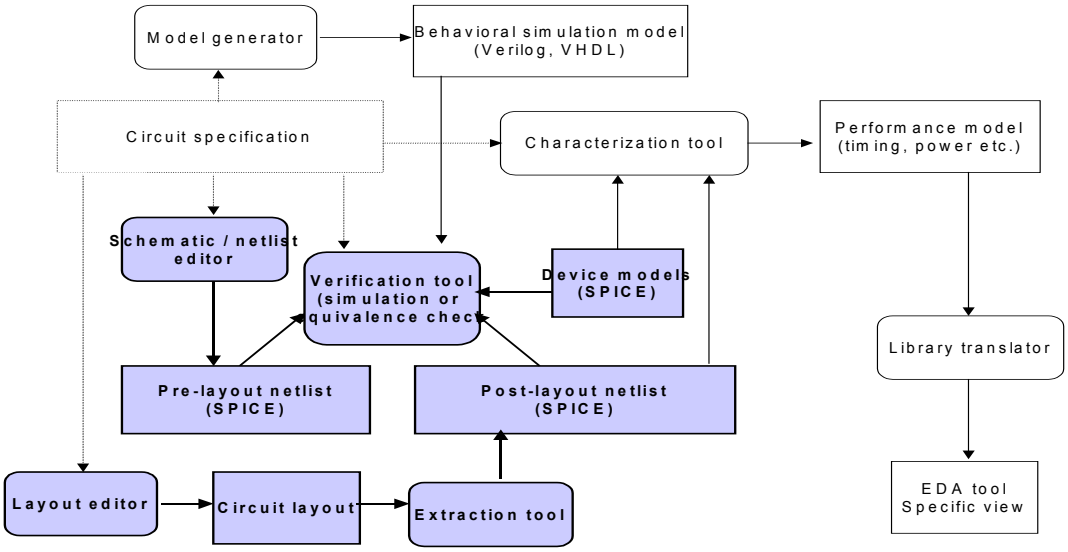
Flow diagrams and information models can be associated with these terms.

Flow diagrams for custom circuit design

A standard for a design kit aims for compatibility between software components and interface files within a design kit. The goal is to achieve interoperability between software components and interface files from different suppliers.

A conceptual flow for custom circuit design is shown below. It shows the steps pertinent for a PDK (emphasized in color) as well as downstream steps. It is important to keep in mind that a PDK does not exist in a closed world but in a context of design applications. The proposed standards should also streamline the interfaces between the PDK and its downstream applications, if possible.

Figure 3 Generic design flow for custom circuit design

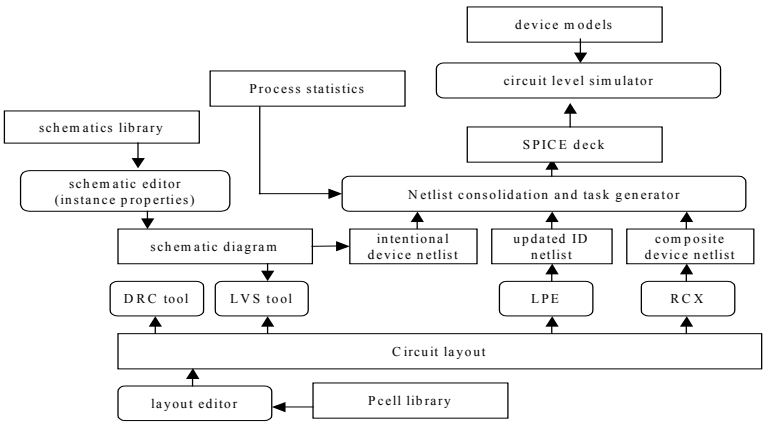


The design activities are driven by a circuit specification, which exists in the mind of the designer. To date, no formal description of design intent is used as part of a PDK. The designer creates a schematic and its equivalent netlist view and eventually a layout, from which another netlist is extracted. Also, the designer may create a behavioral simulation model, either by hand or with the help of a model generation tool. A verification tool or a suite of verification tools verifies whether all the design data, i.e., pre-layout netlist, post-layout netlist and possibly the behavioral simulation models are equivalent, i.e., they describe the same design intent.
Possibly the same simulator used for verification of the netlist will also drive the characterization process. Key inputs to the circuit simulator are device models, the descriptions of which are subject to standardization within this work.

The characterization goal is to create performance models and data that describe the temporal and electrical behavior of the circuit at a more abstract level. Possibly the characterization data are translated into application-specific library formats downstream.
In this flow, some interface files are already represented in industry standard formats, such as SPICE, Verilog etc. Therefore, the digital design domain is “low hanging fruit” in terms of standardization. However, similar formats alone do not guarantee compatibility and interoperability.

Figure 4 shows a more detailed view of the custom circuit design flow at the transistor level. In particular, the emphasized items from Figure 3 are shown in more detail.

Figure 4 Details of a custom circuit design flow at transistor-level



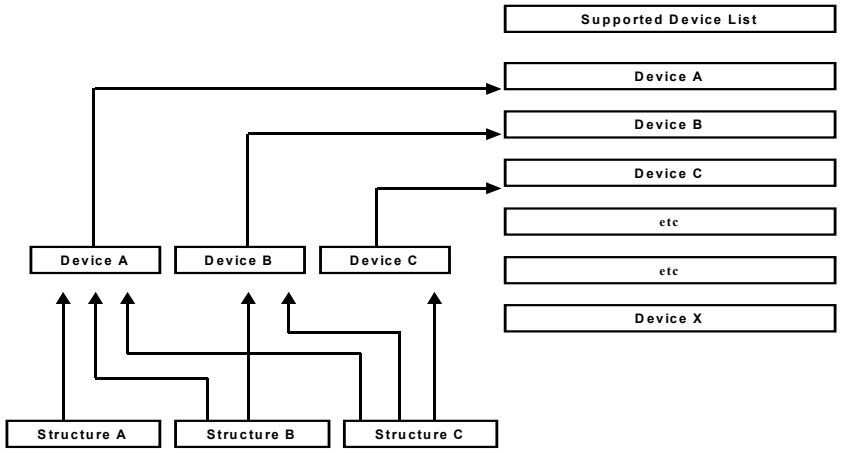
DRC = design rule check (check for correctness of layout rules)
LVS = netlist versus schematic (check for preservation of design intent)
LPE = netlist parasitic extraction (update RC values, preserves nodes in netlist)
RCX = RC extraction (create new nodes as RC parasitics are encountered)

At this level of detail, the non-standardized functions and file representations become more evident.

Information models for custom circuit design

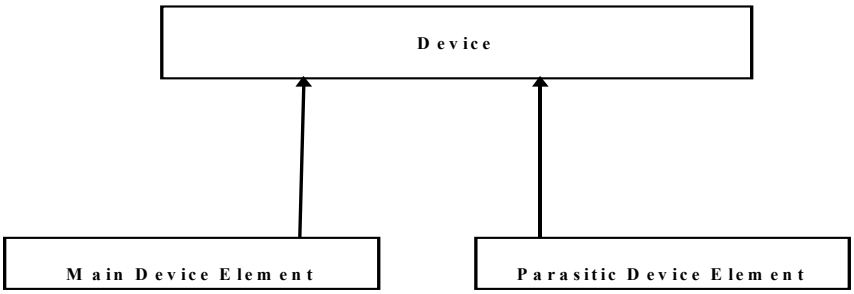
The following figure illustrates the concept of *device* and *supported device list* in the context of a process view.

Figure 5 Process view



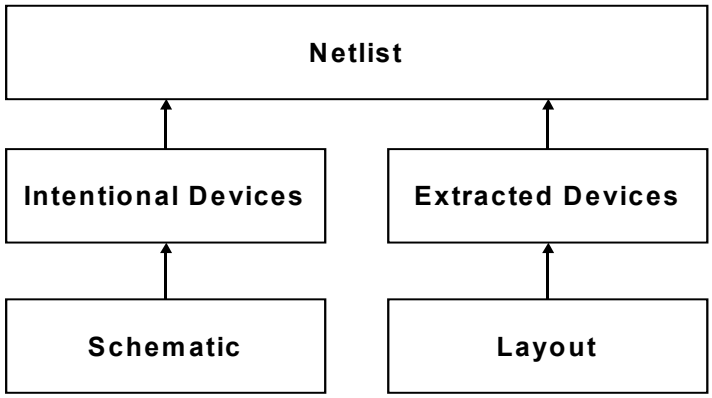
The following figure illustrates the concept of a *core device element* and a *parasitic device element* in the context of a device view.

Figure 6 Device view



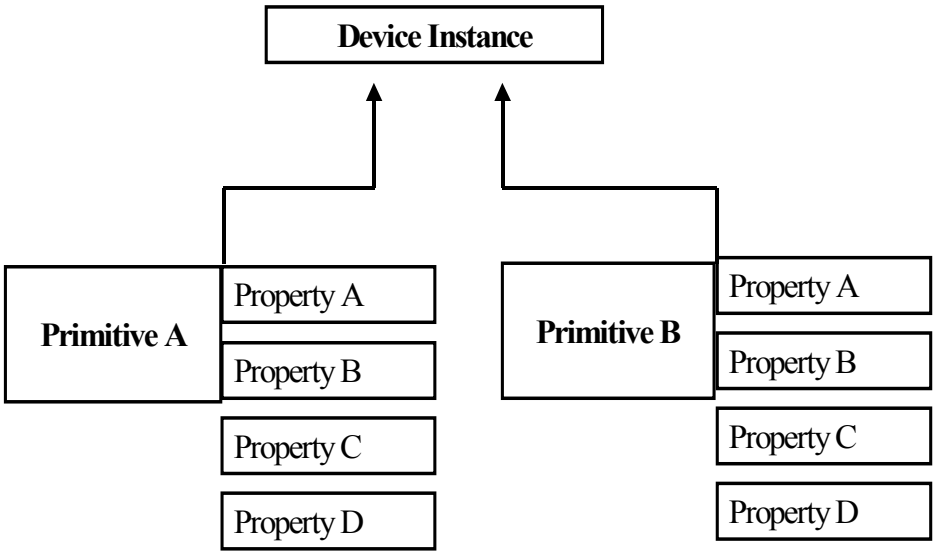
The following figure illustrates the relation between *schematic*, *layout* and *netlist*, *intended device* and *extracted device*, in the context of a design view

Figure 7 Design view



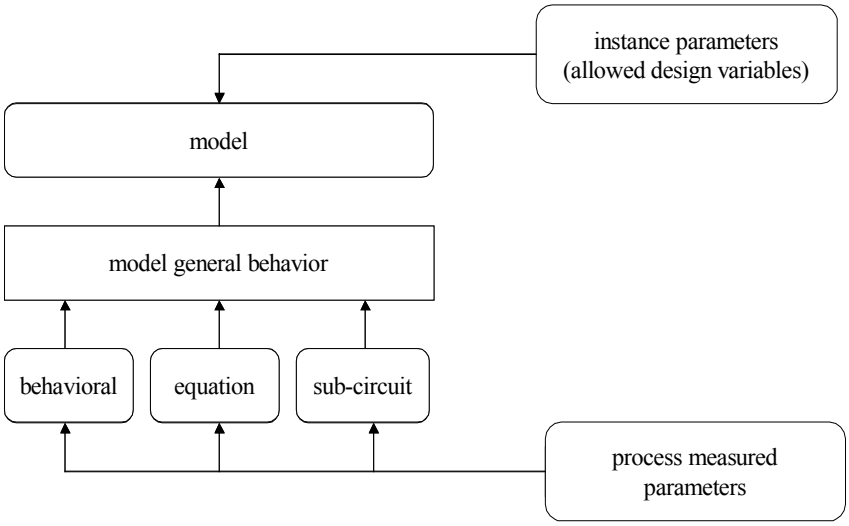
The following figure illustrates the concept of a *device instance*, a *primitive* and a *property*, in the context of a layout view.

Figure 8 Layout view



The following figure illustrates the concept of abstraction or the creation of a model view.

Figure 9 Model view



PDK Contents Definition

The PDK is the software representation of a semiconductor process technology. Following is a list of the items that are required in a baseline PDK. It should provide access to everything a designer needs to know about a process technology in a standardized representation and implemented to work with their EDA environment.

Element	Description	Foundry	Tools	Design	OKI
technology documentation					
device specifications	list of supported devices and their specifications	x			?
layout rules	defines coding standard, layer design rules, device-specific design rules, and sample layouts	x			?
design guidelines	collection recommended best practices on using the device list and process technology	x		x	?
device library					
symbols	schematic representation & properties associated with element in device library		x		x
sample layouts (or p-cells)	example of how to correctly layout & code devices; automated versions are p-cells	x		x	
placement calculations	calculates indirect parameters (callbacks)	x			?
technology file	defines layers and coding standards				
SPICE models	model cards, subcircuits, and behavioral models required to model devices supported by the PDK	x	x	x	
physical verification					
DRC decks	checks layout against design rules	x			?
LVS deck	checks layout against schematic	x			?
LPE deck	updates pre-layout estimates with physical values	x			?
RCX deck	adds extracted devices to the circuit representation	x			?

Candidate Elements for Standardization

The following table is a list of the elements of PDK and associated process technology information that have potential for standardization.

standard	description	notes
standardized device list / process type list:	this is a way of setting the scope of the standards by defining the class (transistor, resistor, capacitor) and type (MOS transistor, poly 1 resistor) of components that have representations covered within the PDK standard. for example, for 'custom digital' we don't need to address LDMOS devices.	
standardized symbol / schematic representations & allowable dialects	the schematic symbol used for a device; this generally would mean specifying the origin, pins, and relative pin locations for a class or type of device; it should also define the basic artwork look and allowable variants (how to represent the two MOS devices in a dual gate oxide technology)	
standardized instance properties & property names:	these are the properties assigned to an instance (specific placement) of a device into a design. basic example: L, W for MOS devices; advanced example would be M for specifying multiple placements or MI for specifying number of fingers in a multi-fingered MOS placement; there are schematic properties and layout properties. this could be expanded to include certain calculations that are associated with the placement of an instance and the property values assigned to that placement.	
standardized layout views (technology file / layers and property names & assignments)	this would involve a consistent cross-foundry methodology for coding the the layers and structures that build up a device. (lots of foundry and IDM input)	
standardized device coding & LVS methods	see above	
standardized simulation views & implementation; e.g. subcircuits):	this is the netlist representation of a device; in this context, it would mean the most complete representation; simulator-specific view would be subsets. there are discussions necessary on representing intra-device parasitics as well as on schematic vs. in-line sub-circuit implementation.	
standardized technology design rule structure & organization:	this would involve a consistent cross-foundry terminology methodology for expressing design rules (layout ground rules) for the layers and structures that build up devices.	

standard	description	notes
standardized layout automation hooks (pins, boundaries):	this is standardization of the locations within or relative to the physical representation of a device where common EDA 'hooks' would be located.	
standardized core device layouts (p-cell templates and features):	standard common set of p-cell features and capabilities for the class and type of a device.	
standardized directory structure and naming conventions	standard location for various elements included in a standard PDK	
standardized QC methodology:	standard check list for ensuring that a PDK is compliant with OK standards and contains no implementation errors affecting it's use a basic custom design flow.	

PDK device representations

Device representations in a PDK can be mapped into a 2-dimensional table, as shown below. The rows define the descriptive representations subject to standardization. The columns define the device categories described by these representations.

Item for standardization	**MOSFET**	**Diode**	**Bipolar**	**Resistor**	**Capacitor**	**other**
Specification	x	x	x	x	x	x
Functional/electrical representation	x	x	x	x	x	x
Physical representation	x	x	x	x	x	x
other	x	x	x	x	x	x

The device categories can be divided into subcategories.

Device category	***Device subcategories***
MOSFET	NMOS (reg,low leak,IO) PMOS (reg, low leak, IO) isolated MOS
Diode	junction diode ESD diode
Bipolar	secondary LPNP secondary VPNP

Resistor	poly resistor implant / diffusion resistor metal resistor
Capacitor	plate-dielectric-plate cap metal comb cap gate oxide cap junction diode cap
other	bond pad

The descriptive representations can be also divided into subcategories.

***Subcategory for* specification**	***Comment***
specification template	i.e. standard device datasheet
intentional devices	
extracted devices	
device-specific design rule format	could be relocated to the process table

***Subcategory for* functional/electrical representation**	***Comment***
symbol	
direct properties (specified/entered)	properties assigned by circuit designer
indirect properties (calculated)	properties calculated
calculation formulae	
simulation view(s)	
model reference	how/where device model reference is included in PDK
simulation model structure	e.g. hierarchical schematic vs. in-line subcircuit representations for intra-device parasitics
LVS reference view	e.g. how core and parasitic devices are managed
ERC reference view	

***Subcategory for* physical representation**	***Comment***
layer coding & assignments	
LVS layout recognition methodology	
device recognition	i.e. how devices are coded / recognized
instance identification	i.e. how two instances of same device type & style are differentiated
example layout	
p-cell feature specifications	

***Subcategory for* other**	***Comment***
isolated device representations	specifying/recognizing isolated devices
multiple device representations	multiplicity factor
fingered device representations	e.g. specifying device aspect ratio
corner model selection	
LPE methods	
RCX methods	
netlist formats	

Using those subcategories, each cell in the table, i.e., an intersection between a row and a column, can be represented as a table in itself, for example, the physical representation of a MOS device.

Physical representation of a MOSFET	NMOS (reg,low leak,IO)	PMOS (reg, low leak, IO)	isolated MOS
layer coding & assignments	x	x	x
LVS layout recognition methodology	x	x	x
device recognition	x	x	x
instance identification	x	x	x
example layout	x	x	x
p-cell feature specifications	x	x	x

The detailed 2 dimensional table will allow to examine proposals and technology donations for suitability and completeness.

PDK technology rule representations

Technology rule representations in a PDK can be mapped into a 2-dimensional table, as shown below. The rows define the descriptive representations subject to standardization. The columns define the device categories described by these representations.

Item for standardization	**Layer class A**	**Layer class B**	**other**
Layer information	x	x	x
Design rules	x	x	x
Layer & structure specifications	x	x	x
other	x	x	x

Please explain Layer class A and B.

The descriptive representations can be divided into subcategories.

***Subcategory for* Layer information**	***Comment***
layer types & purpose	e.g. mask, drawing, helper, boundary, dummy, etc.
layer sourcing	e.g. input (drawn) versus calucated/derived
layer names	naming convention (and/or mapping)

***Subcategory for* Design rules**	***Comment***
design rule terminology	e.g measurement case definitions, e.g. overlap, overhang
design rule illustrations	e.g. standard illustrations for types of layers
design rule formats (geometic)	e.g. standard table organization and structure or cases & values affecting layer geometries
design rule formats (electrical)	
design rule formats (manufacturability)	e.g. standard table organization and structure for process antenna rules, pattern density rules
design rule formats (reliability)	e.g. standard table organization and structure for electromigration rules
treatment of scaling (at tapeout)	for technologies defined with a "shrink factor"
design rule information location(s) in PDK	i.e. where & how design rules are represented in the PDK (rules files)

***Subcategory for* Layer & structure specification**	***Comment***
layer & structure electrical characterisitcs specifications (format)	e.g. resistivity, capacitance, etc.
layer & structure physical characteristics specification (format)	e.g. thickness
layer & structure electrical characteristics PDK location(s)	i.e. where & how layer & structure electrical characteristics are stored in PDKs
layer & structure physical characteristics PDK location(s)	i.e. where & how layer & structure physical characteristics are stored in PDKs

***Subcategory for* other**	***Comment***
structure representation & support	e.g. via, contact & other sub-device items
coding grid methodology	e.g. use of edge vs. centerline
database unit standards	
treatment of design size adjusts	e.g. how differences between drawn and on-silicon dimensions are handled
non-rectilinear representations	e.g. grid coding methodology for circles and arcs
layer visualization	e.g. color or stipple pattern
representation of (design) mask options	e.g. metal mask options for different designs
DRC deck organization & structure	

PDK Qualification

The goal of PDK qualification is to provide guidance to allow for PDKs to be qualified to the OpenKit standard. Currently, this section provides examples and proposes some Quality Assurance (QA) items on the front-end. QA on the back end is an open issue at this point.

QA – Front-end

For each device:

- Device symbol look (size, layout, #pins, pin locations etc.) obeys OKI definitions
- Device allows input of appropriate parameters as defined by standard (e.g. width/length/fingers/multiplicity) for mosfets, w/l/r for resistors etc.)
- Parameter input should allow numeric or string (so user can enter a variable name for the parameter rather than a number, needed for optimization)
- Parameter names are as specified in standard (e.g. “width” or “w”?)
- Device calculates standardized parameters (e.g. ad/as/pd/ps for mosfets) and does it correctly
- Device netlists correct parameters with correct value and syntax for all tools
- Device should not allow entry of incorrect data (e.g. should have checks in place so that user can’t enter a device size that disobeys design rules).
- Devices should display tool specific information correctly (e.g. current, voltage from simulators)

QA – Front-end test suite Create schematic(s) containing one of every device in the PDK instantiated with default values

- Create schematic(s) for each device and for each parameter. Need multiple device instantiations that vary the parameter from min to max with appropriate increments
- Create schematic(s) for each device with parameters set to values used for in layouts for model extraction (compare performance of models to silicon)
- Netlist each test schematic for every supported tool:
    - Verify that all parameters are output correctly (value, syntax etc.)
    - Run each netlist in the tool to make sure no errors are detected and document all tool versions which were tested and results (include replay file to regenerate results if supported by tool) – this will also test syntax correctness of models
    - Include this “golden” netlist with the PDK so user can validate against it

QA – Models Model should contain no syntax errors

- Model is defined for all supported values (e.g. if binning then should have bins covering min to max param ranges) of device model parameters
- Model supports standardized characterization effects (1/f noise, defined threshold ranges, temp coeff, voltage coeff etc.)
- Model documentation includes details of silicon devices upon which model is based (sizes, types, variations etc.)
- Model documentation includes measured data from silicon
- Model is used to generate I/V curves to validate there are no undefined modeling regions. I/V curves are supplied as part of design kit documentation
- Model validation occurs across all tools with correlation between all tools and tested/supported tool versions are documented with model release
- Standardized corner models are supplied and model is tested across all corners

Version/Release control Provide design kits in a way that allows user to "lock" and maintain a given release (e.g. when tapeout a chip want to retain the design environment as it was at tapeout)

- Provide documentation about all tested & supported tools and the versions which were used for testing
- PDK supplier must use revision control software (e.g. rcs, sccs, cvcs, DesignSync, etc) for all objects and provide revision history of all objects to user at each release
- If model/PDK updates are separate then model to PDK version correlation must be provided
- PDK release should be complete and include all "pieces" valid at time of release (PDK libraries, associated code, models, etc)

Roadmap

Technology Reference

The benchmark process technology definition that will be used to drive the initial set of standard will be a 180 nm BiCMOS process technology with copper interconnects. This technology is selected to contain a broad range of supported devices to use is setting standards. The initial focus will be on custom digital design using CMOS devices, but the broader set of devices provides a roadmap and larger technical context for considering standardization.

Standardization / Release Roadmap

Rev	Target	Design Class	Technology Node	Device Categories Covered

Rev	Target	Design Class	Technology Node	Device Categories Covered
1.0	06/04	CMOS custom digital	180/130 nm CMOS	NMOS, PMOS, poly resistor, MOS capacitors, metal capacitors, metal resistors, diodes
1.1	12/04	Analog CMOS and power	180/130 nm BiCMOS	poly capacitors, precision devices, drain-extended CMOS, DMOS, BJT (power)
1.2	06/05	RF BiCMOS	180/130 nm BiCMOS	BJT (signal), inductor, varactor
2.0	12/05	CMOS custom digital	90 nm CMOS	See 1.0

The roadmap is intended to track the ITRS roadmap with 3 design class / device category releases per technology node.