Si2 Low Power Glossary™

v1.1

20 July 2009

Published by
Silicon Integration Initiative, Inc. (Si2™)
9111 Jollyville Road, Suite 250
Austin TX 78759

ISBN: 1-882750-40-3

The following description outlines an approach adopted by much of the industry regarding definitions of low power terms. This information is provided only as a description of the current state of the art, and does not constitute a recommendation or contribution on the part of Si2 or any Si2 initiative or Member regarding implementation approaches. This information is not a standard or a specification that has been approved or adopted by Si2 or by any Si2 initiative or Member, including without limitation Si2's Low Power Coalition (the "LPC"), and has not been subjected to a "Call for Patents" by the LPC. All information in this description is provided "AS IS" without warranty or representation of any kind including without limitation any warranty or representation of merchantability, fitness for particular purpose or non-infringement. No information in this description shall obligate Si2 or any Si2 Member to grant licenses under any patent rights controlled by Si2 or such Member with respect to any implementation or other use of this information.

The requested document describing the LPC Glossary and all materials and information therein, are provided as is and without warranty of any kind. The authors, editor, publisher, and contributors specifically disclaim warranties of any kind whether express or implied, with regard to any material contained herein, including, without limitation, any warranties of title, non-infringement, merchantability, or fitness for a particular purpose or arising from course of dealing or usage in trade are made or shall apply.

In no event shall the authors, editors, publisher, or contributors, be responsible or liable for any loss of profit or damages, direct or indirect, including but not limited to, special, incidental, punitive, indirect, or consequential damages of any nature arising from or relating to the specification or any materials or information therein, even if advised of the possibility of such loss or damages.

Notice: Attention is called to the possibility that implementation of this specification may require use of subject matter covered by patent rights under which a license may be required. Si2 shall not be responsible for identifying patents or patent applications for which a license may be required to implement an Si2 specification or for conducting inquiries into the legal validity or scope of those patents that are brought to its attention. By publication of this document, Si2 takes no position with respect to the existence or validity of any patent rights.

A patent holder or patent applicant has, however, filed a statement of willingness to grant a license under these rights on reasonable and nondiscriminatory terms and conditions to applicants desiring to obtain such a license. Si2 makes no representation as to the reasonableness or nondiscriminatory nature of the terms and conditions of the license agreements offered by any such holder(s). Further information may be obtained from Si2 upon request.

Trademarks: Where manufacturer or vendor designations claimed as trademarks are used herein, and the publisher or authors were aware of the trademark claim, such designations have been printed in initial caps or all caps.

Restricted rights: Use, duplication, or disclosure by the Government is subject to restrictions as set forth in FAR52.227-14 and DFAR252.227-7013 et seq. or its successor.

Contributors

Andres Teene	LSI
Anmol Mathur	Calypto
Bob Carver	Si2
Dave Ollson	NXP
David Hathaway	IBM
Dirk Siemer	Chipvision
Gary Delp	SPIRIT
Geoff Mole	NXP
Jerry Frenkil	Sequence
John Biggs	ARM
Judith Richardson	AMD
Nagi Naganathan	IBM
Qi Wang	Cadence
Rachida Kebichi	AMD
Sheela Shreedharan	Viragelogic
Steve Urish	IBM
Tom West	Chipvision

Purpose

This collection of definitions has been produced to enhance communication related to power-aware design and to promote industry-wide standardization. As such, wide distribution and use is encouraged. The LPC actively solicits inputs from all of the various stakeholders in the EDA industry, both inside and outside of the LPC to enhance this document. The LPC Glossary Forum is available for suggestions of new terms or changed definitions.

ESL Phase

The ESL phase of the design process embodies the transformation of a High-level System (Product) specification into System Architecture. This phase provides a platform for system and block level power, performance, and area optimization and includes:

- Power and performance analysis in both time and space domain (frequency and data width scaling),
- Quantitative tradeoffs for hardware or firmware partitions,
- Selection of large IP components (e.g, processor cores),
- Facilitation of early design closure to meet system specification requirements,
- Development of comprehensive power management features,
- Produce mixed simulation platforms for both RTL and abstract models,
- Creation of a power budget for subsequent flow phases,
- Power analysis at scalable runtime based on accuracy requirements (transaction or cycle accurate),
- Minimum impact to the existing ESL design environment,
- Definition of clock frequencies and interfaces between potentially asynchronous clock domains,
- Architectural (rather than inferred) clock gating definition.

Design Phase

The Design Phase is the detailed selection, modification and coding of RTL descriptions of the major subsystems within the architecture. In addition, the subsystems are interconnected during the Design Phase. Each sub-system is mapped into one or more power domains and the modes of operation for the design are refined into specific nominal operating conditions for each domain. Domains that must preserve state (or

some portion thereof) are identified. Rules for the treatment of inter-domain signals are defined. RTL power analysis and linting is performed with static and dynamic power targets for each power mode. The power modes are decomposed into dynamic and static power budgets and constraints for each power domain at its appropriate operating condition. The RTL for the chip-level power management unit is created during this phase.

Implementation Phase

The mapping of the RTL and power description file into physical devices that can realize the design intent. Implementation tools must ensure that all constraints are satisfied. In particular, implementation tools may generate additional constraints that must be satisfied by subsequent tools.

ESL Description(s)

Algorithmic View (not abbreviated as AV)

Many would consider the output of Matlab® or possibly SPW (Signal Processing Worksystem) (without the HDL interface) to play in this space. There is no concept of hardware, no concept of bus structure, no concept of a processor or software. This is strictly the algorithm written in a programming language in the most efficient form possible. It may be possible to synthesize this with behavioral synthesis if this represents strictly a piece of IP (e.g. an FFT algorithm), but not if it represents an algorithm that will be spread across an SOC. There is no concept of RTL structures such as cells, modules, and leaf levels, at this level of abstraction.

Programmers View (PV)

The Programmers View is a view often derived from the Architects view (below). This represents a complete SOC as a programmer would need it. Performance is the most critical concern. Programmers demand tens of MHz performance. Often stubs or traffic generators are used for peripherals. There is no bus structure, no arbitration, or protocol although the processor does see the peripherals through the memory mapped space (routing). At the very least, the peripherals must have the complete mapped register interface. There is a refined view called PV-T which includes some annotated bus timing for major bus events such as request, grant, address valid, data valid, etc. There is no concept of RTL structures such as cells, modules, and leaf levels, at this level of abstraction.

Architects View (AV)

The Architects View is the same as the PV with accurate annotated bus timing. This is often the same PV model with PV to (bus protocol) transactors inserted using a transaction level bus model. This view is used by architects to accurately perform interconnect analysis, bus bandwidth analysis, processor performance, data throughput etc.. Often stubs or traffic generators are used as peripherals. “Lumped” cycle accuracy is achieved at the transaction level through lumped annotated timing. "sc_signals" are rarely used except for an occasional sideband signal like an Interrupt. There is no concept of RTL structures such as cells, modules, and leaf levels, at this level of abstraction. Interrupts are considered one of the few things that need timing (i.e. threads and “wait”) in the models; the rest of the timing gets annotated to the bus.

Verification View (VV)

The Verification View is a less commonly used ESL view. This view is used when Architects want to test RTL through either an HDL co-simulator or HDL that has been translated to RTL-level SystemC. The AV model would have transactors where necessary to achieve the cycle-accurate interface to the cycle-accurate IP. This mode is full of slow cycle-accurate SystemC constructs such as sc_signal. This view represents an abstraction change from the above views as this view generally represents a separate model that can be achieved by switching on timing features from the above views.

Behavioral Description

A Behavioral Description is a description of the design in terms of desired behavior. The behavioral description can be decomposed into a data-flow graph and a control-flow graph; however, the nodes in this graph have not yet been allocated to synthesizable elements or scheduled across particular clock cycles.

RTL Description

An RTL (Register Transfer Logic) description describes the design as a hierarchical set of synthesizable RTL modules, usually Verilog or VHDL. The power file may augment the RTL description either explicitly or implicitly adding additional design objects to support the power intent.

Net list/Logic/Gate Description

This type of description represents the design as a set of interconnected instances. All computational elements have been previously elaborated by synthesis. The power file may augment the design description either explicitly or implicitly adding additional design objects.

Physical Level Description

The physical level description describes the design as a set of cell placements, wire routes and physical shapes necessary to realize the design in silicon. All design objects in the power file will have been explicitly merged into the physical description. The physical hierarchy may be different than the logical hierarchy.

Power Design

A power design is a self-contained package of power design intent for a module, including any power domain and rule definitions. More than one power design may be specified for a module, and different instances of the module in the design hierarchy may reference (be bound to) different power designs. A power design is required only for the root module of a design hierarchy. A module instance not bound to a power design will get its power intent from its closest ancestor in the design hierarchy for which a power design is specified.

Power Domain

A power domain is a named collection of design elements, instances, pins and ports determined by a user specification. The semantic requirement is that all of the elements share the same primary supply set (1). A power domain is also sometimes used in CPF as a proxy for its primary supply set, see primary power domain and secondary power domains.

At any given time, each of the supply set functions of the elements of a power domain must resolve (be connected to) the same supply net or to equivalent supply nets.

Objects within a power domain may be associated with other supply sets in addition to the primary supply set and take power from those supply sets. Examples of these include level shifters which have input and output supply sets.

Power domains (that is the collection of elements referred to by a named power domain) may be combined into a composite power domain.

At the physical level, a power domain corresponds to one or more power islands.

At the logic level, a power domain contains:

- A set of logic instances that correspond to the physical instances of this power domain

- Optionally, a set of special cells such as level shifter cells, state retention cells, isolation cells, power switches, always-on cells, or multi-rail hard macros (such as, I/Os, memories, and so on) that correspond to the physical implementation of these cells in this power domain (2,3).
- At the RTL level, a power domain contains:
- The computational elements (operators, process, function and conditional statements) that implement the logic instances in this power domain
- Optionally, a set of special instances as described above.

Notes:

1. In special situations in IEEE 1801™ 2009, using the connect_supply_set -elements {}, the primary supply set of an element in a power domain may be reassigned to a supply set that is not the default primary supply set of the power domain. CPF does not allow this flexibility.

2. Because objects within a power domain always share the same nominal operating conditions, level shifters or isolation cells are never needed on signal connections between objects within the same power domain.

3. Two objects should not be inferred to be in the same power domain merely because they always share the same nominal operating conditions. Some explicit action by a designer or tool must be taken to merge power domains or place objects into a common power domain.

Composite Power Domain

Power domains (that is the collection of elements referred to by a named power domain) may be combined into a composite power domain. The composite domain contains all of the elements in each of its component or child domains. At any given time, each of the supply set functions of the primary supply sets of the child domains and the primary supply set of the composite domain must resolve (be connected to) the same supply net or to equivalent supply nets, otherwise there is an error in construction.

Supply Set

A supply set is a (named) collection of supply set functions that can be used to power design elements, domains and/or rules. A supply set will by default include the functions of power and ground. Power states may be associated with supply sets, and simstates may be associated with the the power states of a supply set.

Notes:

4. A supply set can be referred to by using its name or supply set handle

5. Named supply sets are defined in the current scope, and can be used as placeholders, allowing the specification of supply net connection later in the design process. Several of the objects in the data model have defaults names for their supply set connections. The names, or handles can be used wherever a named supply set might be used.

6. Examples of using supply set handle for a named Power Domain PD1:

    1. PD1.primary will refer to the primary supply set of that domain.
    The supply nets of a domain's primary supply set are implicitly connected to any design element from the logic hierarchy that is within the extent of the domain if the element has no supply ports defined on its interface.

    2. PD1.retention will refer to the default retention supply set of that domain.
    If no more specific supply set is specified, this supply set will be used for logic which requires a retention supply.

    3. PD1.isolation will refer to the default isolation supply set of that domain.
    If no more specific supply set is specified, this supply set will be used by isolation rules of this domain.

    4. For a retention rule RR1 defined in Power Domain PD1 as PD1.RR1:
    PD1.RR1.supply is the retention supply set for the retention elements inferred by the RR1 rule.

    5. For an isolation rule ISO1 defined in Power Domain PD1 as PD1.ISO1:
    PD1.ISO1.supply is the supply set for the isolation elements inferred by that rule.

Supply Set Function

A supply set function is implemented by a supply net for a specified purpose within a supply set. Within a supply set, any number of supply set functions may be defined. The following names are reserved for predefined supply set functions and may be used in any supply set definition.

1. **power** is the supply net which provides the power function of the supply set and is connected to ports having the pg_type primary_power.

2. **ground** is the supply net which provides the ground function of the supply set and is connected to ports having the pg_type primary_ground.

3. **pwell** is the supply net which provides the pwell bias of the supply set and is connected to ports having the pg_type pwell.

4. **nwell** is the supply net which provides the nwell bias of the supply set and is connected to ports having the pg_type nwell.

5. **deeppwell** is the supply net which provides the deeppwell bias of the supply set and is connected to ports having the pg_type deeppwell.

6. **deepnwell** is the supply net which provides the deepnwell bias of the supply set and is connected to ports having the pg_type deepnwell.

Notes:

1. As above with supply sets, the supply set functions can be referred to with symbolic names

    - PD1.retention.supply.power will refer to the default retention supply power net of that domain.
    - PD1.RR1.supply.power is a reference to the retention supply power net for the retention elements inferred by RR1 rule. Note well that PD1.retention.supply.power and PD1.RR1.supply.power may be equivalent supply nets.

Base and Derived Power Domains

For two power domains X and Y, where the primary power supply of X provides power to Y through a power switch network, regulator, or other means, the following relationships are defined:

- X is a base of Y
- Y is derived from X

Note: A power domain may have multiple bases.

Note: A power domain X may be both a base power domain for Y and be a derived power domain from another power domain W.

Primary Power Domain / Default Power Domain

The power domain X of a module instance may also be referred to as the primary or default power domain of the module, because all cell instances in the module (including those within child module instances) which are not explicitly placed in other power domains will also be in power domain X. This also means that the primary supply set of X will be the primary supply set of these contained cell instances.

Secondary Power Domain

A power domain X is a secondary power domain of an instance or module if the primary supply set of domain X supplies a special purpose power (in particular for isolation, retention, or always on cells) to the instance or the instances of the module.

Primary Supply Set

The primary supply set of a power domain provides implicit power connections to cells within the power domain that do not have explicit power connections.

The primary supply set of a cell (e.g., a level shifter) in a power domain implemented using standard cells will be implicit power, ground, and possibly other connections created by placement of the cell within a standard cell row, if such implicit connections exist.

Power Island

A power island is a geographically bounded collection of instances, all from the same power domain. Each of the supply set functions of the primary supply sets of all instances within a power island must be implemented by the same supply net.

Notes:

1. A power island contains:

    - A set of physical instances that correspond to the logical instances of the associated power domain.
    - Optionally, a set of special instances such as level shifter cells, state retention cells, isolation cells, power switches, always-on cells, or multi-rail hard

macros (such as I/Os, memories, and so on) that may have other power connections in addition to the primary power supplies of its power domain.

2. Since they are geographical, power islands are used in the physical level description to "realize" power domains. A power domain may be implemented as multiple, possibly disjoint power islands; thus both, one-to-one or many-to-one relationships between power islands and power domains are possible. However, in the case of a many-to-one relationship the power islands must be constructed such that signals may pass between any two power islands of the same power domain without the addition of any special power interface logic (e.g. level shifters/isolation devices).

Equivalent Supply Nets

Equivalent supply nets of a power domain are those nets which can be interchanged with each other in terms of supply function, without any change in the behaviour of the circuit.

Notes:

1. Supply nets may be declared to be equivalent. Part of the verification of a specification is to ensure that those supply nets declared and used as equivalent do in fact produce the identical effect when interchanged.

2. At the physical design level, supply nets declared to be logically equivalent may differ in voltage drop, location, etc.

Power Mode

A power mode is a specific power configuration of the design in which each power domain in the design has a specific nominal operating condition and associated clock frequency.

Power Mode Control Group

A set of power domains with an associated set of power modes and mode transitions that apply only to this group. A power mode control group can contain other power mode control groups. The intent of this is to allow modeling of hierarchical power intent. A set of power modes for a particular power mode control group are mutually exclusive, but power modes within different power mode control groups are *not* mutually exclusive. Thus a design may simultaneously be in one power mode from each power mode control

group defined. A design incorporating different power mode control groups may define specific composite power modes (allowed combinations of power modes from its constituent power mode control groups).

Nominal Operating Condition

The Nominal operating condition specifies a set of voltages that will be applied to the primary power supply sets for one or more power domains.

Notes:

1. Once it is linked to a power domain, the nominal operating condition specifies a logical state for the associated power domain.

2. Operating corner should be used to specify process and temperature information in addition to the voltages for implementation.

Issues / questions:

- Should we include current limits and/or source impedances for a power supply here or elsewhere?
- Need a reference to the simstate to indicate the logical state specified by nominal condition

Mode Transition

A mode transition describes a transition between two different power modes.

Operating Corner

An operating corner is a specific set of process, voltage, temperature values under which the design must be able to perform.

Analysis View

An analysis view associates an operating corner with a power mode for which timing constraints are specified. The set of active views represent the different design variations (MMMC, that is, multi-mode multi-corner) under which correct operation of the design must be implemented and verified.

Rules Section

Rule

A rule identifies those design objects that must be modified and/or inserted to support proper operation under the various power modes of the design. Rules can specifically identify a design object by name or use a pattern to generically identify the design objects

Issues / questions:

- The section head (*Rule*) above should not be considered one of the glossary definitions, but instead is just a grouping of the following definitions. This is to avoid confusion with the term library "rules" (*e.g.*, .lib). We should clarify this or propose an alternative generic term.

Inter-domain Rule

- An abstract rule that identifies the pins (drivers or receivers) attached to inter-domain signals.

Issues / questions:

- Is there (or should there be) a separate rule type defining how signals between two domains should be buffered (e.g., when passing through a region that is not in either domain), or should it be included in one or both of the following rule types?

Isolation Rule

An isolation rule is an inter-domain rule that identifies pins that must be isolated on inter-domain signals. An isolation condition is given in the rule that specifies when the isolation logic should be enabled. Typically this is used for drivers contained within a power domain that may be switched off. The rule may optionally specify the logic value that the signal should be driven to when isolation is in effect, the type of isolation cell to be added between the driver and the receivers and the power domains that will contain the inserted isolation cell.

Level Shifter Rule

A level shifter rule is an inter-domain rule that identifies the pins attached to inter-domain signals that must be level shifted. The rule may optionally specify the type of level shifter cell to be added and the power domain that will contain the inserted level shifter cell.

Power Switch Rule

A power switch rule is a rule that specifies the location and the type of power switch cell to be added to the design and the condition for when to enable the power switch.

State Retention Rule

A state retention rule is a rule that identifies the sequential storage elements to be replaced with state retention cells and the conditions for when to save and restore their states.

Design Objects

A design object is an instance, pin, port or net in the design description with a name that uniquely identifies it.

Design

The **design** is an ancestor for all instances in the design hierarchy. It is sometimes refered to as the top-level module.

Issues / questions:

- Should we include something like "The top level module for the current design and analysis operations?" It is important (as we keep saying) to enable hierarchical design, and in this case the top level depends on what you're doing.

Instance

An instance is instantiation of a module or cell (or computational element at the RTL level) within the design hierarchy. An instance belongs to one and only one power domain. However the pins of an instance may optionally belong in a different power domain from that of the instance (this implies that the instance will have multiple power connections made to it). Unless specified otherwise an instance exists in the same power domain as its parent. In OpenAccess an Instance is represented by oaModInst, oaOccInst and/or oaInst.

Module

A module is a synthesizable RTL description in the design, typically described in either Verilog or VHDL. In OpenAccessa Module is represented, if at all, by an oaModule.

Net

A net is a connection between instance pins and/or ports at one level of the design hierarchy. In OpenAccess a Net is represented by oaModNet, oaOccNet and/or oaNet.

Pin

A pin is a connection point on an instance where a logic/power/ground signal/net may be attached. Unless specified otherwise a pin is in the same power domain as its instance. In OpenAccess a Pin is represented by oaModInstTerm, oaOccInstTerm and/or oaInstTerm. Pins are logical entities and should not be confused with the physical entity oaPin

Port

A entry point to or exit point for a module or cell where a logic/power/ground signal/net may be attached. A port on a module or cell has an associated pin on the instance that references that module. In OpenAccess a Port is represented by oaModTerm, oaOccTerm and/or oaTerm.

Cell

A cell is an element in a library (*e.g.*, a standard cell). This may include large hard IP blocks (*e.g.*, processor cores, memories, etc.). In OpenAccess a cell may be represented as an oaDesign.

Library

A library is a collection of cells described in a particular format (*e.g.*, Liberty .lib).

Library Group

A list of libraries characterized for two or more operating conditions. All libraries in a group must have the same cells. A library group can be used in a DVFS design to interpolate power or timing data for any operating conditions.

Library Set

A set (collection) of libraries or library groups. By giving the set a name it is easy to reference the set when defining nominal conditions or operating corners. The same library set can be referenced multiple times by different operating corners.

Scope

A scope identifies a namespace within the design hierarchy for resolving relative references to design objects.

Leaf Instance

A Leaf Instance is an instance that is not decomposed into a lower level description for the purpose at hand. These instances form the leaf nodes in the Design Hierarchy. Leaf instances must be instances of library cells, I/O pads, or a hard IP block (*i.e.*, a leaf instance has no children for the purpose at hand).

Special Library Cells for Power Management

Always-On Cell

An Always-On special cell located in a power domain that continues to operate even when the power domain it is located in is shut off. "Always" in this context may be relative; an "Always-On" cell will have an additional supply that remains "on" when the normal supply for the power domain in which it resides is "off", but it is still possible that this additional supply will also be turned "off" under some circumstances (*e.g.*, a deeper sleep mode).

Isolation Cell

An Isolation Cell is a cell used to isolate signals between two power domains where one power domain maybe switched off while the other power domain continues in normal operation. The most common usage of such cell is to isolate signals originating in a power domain that is being switched off, from the power domain that receives these signals and that remains switched on.

Level Shifter Cell

A Level Shifter Cell is a cell used to pass data signals between power domains operating at different nominal operating conditions.

Power Clamp Cell

A Power Clamp Cell is a special diode cell to clamp a signal to a particular voltage.

Power Switch Cell

A Power Switch Cell is a cell used to connect and disconnect a power supply to a power domain.

State Retention Cell

A State Retention Cell is an always on cell such as a flip-flop or latch that can retain its logic state when the primary power supply to the cell is shut off. The logic state is preserved subject to a save and restore signal.

Glossary - Work in Progress

Clock Domain

Power Partition
