Low Power System-on-Chip Design

Advanced Power Modeling Support in today’s EDA Flows

Petri Solanti, CAE
Synopsys Finland Oy
Part 1 : Power Analysis with Virtual Platforms

- What is a Virtual Platform
- Power Modeling & Estimation goals
- Power Modeling Support
- Clock Modeling
- Voltage Modeling
- Power Estimation
- Dashboards
What is a Virtual Platform?

• Fully functional software model of complete systems
  • SoC, board, I/O, user interface
• Executes unmodified production code
  • Drivers, OS, and applications
• Runs close to real time
  • Boots OS in seconds
• Highest debugging efficiency through full system visibility and control
  • Supports multi-core SoCs debug
Virtual Platform – Closer Look

- Virtual I/O
- User Interface Emulation
- High-Level, High-speed C/C++/SystemC Models
- Fast Instruction-Accurate Simulator
- System IO
- Board-level
- System-on-Chip
- CPU
- Func TLM Bus
- Functional Peripherals
- Transaction-level Interfaces
- Graphical Peripheral Models

Simulation Infrastructure

Virtual I/O & User Interface
Power Modeling & Estimation Goals

- **Increase visibility** into global system state & individual power/clock domains

- **Architectural power trade-offs**
  - Explore performance vs. power trade-offs
  - Test drive new power scheme with “system software” load

- **Power-related software development**
  1. Perform software power optimizations by providing (relative) power consumption estimations
  2. Enable development of power management software
  3. Provide insight into system power consumption, when running *actual* system software
Power Modeling Support

- The following power support is added to a TLM platform:
  1. Power management modeling
     1. Clock modeling – gating, scaling freq(t), …
     2. Voltage distribution - power domains, scaling V(t), …
     3. Power state control – power state sequencing (power down, retention, …)
  2. Power estimation equations – evaluated at run-time
  3. Dashboards – clocks, state, voltage, power

- Applies both to PV & PVT level modeling
  - PV
    - Instantaneous power consumption, at each point in time
  - PVT
    - Supports trade-off of performance vs. power
    - Graphs: Power(t) & Energy(t)
    - Improved accuracy for accounting for (memory) transactions power contribution
System-level Power Exploration
SW-Driven Power Analysis & Optimization

Start with functional-accurate TLM System Model

Run System Software

Specify Voltage Domains

Add Clock Distribution

Add Power Estimation Equations

Add Power Management Schemes

Analyze Power Results

- Domain Power Consumptions
- Device Energy
- Power Events

Optimize System

This material is property of Synopsys Inc. Using this presentation or individual slides without permission from Synopsys is prohibited.
Clock Modeling

- **Functional clock modeling**
  - Models functional operation of the clock controller, including:
    - Clock distribution
    - Control over peripheral clock gating
    - Registers, to model software control over clock frequencies

Signal is used to transmit value of clock frequency from controller to peripheral; does not model actual clock waveform
Voltage & Power State Control Modeling

- **Functional models of:**
  - **On-chip Power Manager (SoC)**
    - Control of (internal) voltage distribution & domains
    - State control & sequencing
  - **Power management chip (PMIC)**
    - Voltage scaling of SoC
    - SoC I2C control interface, power sequencing, LDO regulators control, DC/DC convertors, …
Power Estimation - Concepts

Parametrizable power model, consisting of:

1. Component power characteristics
2. Component power calc. equations
3. Power accumulator
Power Estimation – Component Characterization

- **Power parameters**
  - Components are characterized by a set of representative power parameters (‘kernels’)
  - Used in power equations to calculate power
  - Flexible to support specific component characteristics
  - Interactively changeable by user

- **Source**
  - Power consumption numbers are delivered by semiconductor company
  - Based on (1) budget planning, (2) estimations, (3) measurements
Power Estimation – Component Characterization

- **Examples**: OMAP2420 (PV model)
  
  1. **CPU**
     - Power active (mA / MHz)
     - Power dormant (mA)
     - Power inactive (mA)
     - Power shutdown (mA)
  
  2. **Peripherals**:
     - Power *clock off* (mA)
     - Power *idle* (mA / MHz)
     - Power *typical* (mA / MHz)
     - Power *maximum* (mA / MHz)
  
  3. **On-chip Memories** (RAM / ROM)
     - Power *clock off* (mA)
     - Power *idle* (mA / MHz)
     - Power *read transaction* (mA / MHz)
     - Power *write transaction* (mA / MHz)
Power Estimation

- **System power estimations expressions**, contain:
  1. Component power characteristics
  2. Power state of APLLs & DPLLs \{ off, on \}
  3. Power state of domain \{ off, ret, on \}
  4. Voltage applied (to domain)

- At run-time power estimation expressions are evaluated on the fly, as triggered by user requests
  - Complements functional component model
  - Leverage power state & frequency modeling of func. component model
  - Expressed in C code (Magic-C or C++)
Power Estimation (cont’d)

- Expressions can be linear, or more complex, depending on:
  - Component type
  - Data / characteristics which can be measured / estimated

- Fixed modeling APIs for voltage, frequency & power state updates, and reporting to accumulator
Power Estimation - Example

**ARM1136 (OMAP2420)**

**ARM MPU Power Estimation Component**

- MPU Voltage (Volt)
- MPU Clock (frequency (MHz))
- MPU Power State (on, off, …)

This material is property of Synopsys Inc. Using this presentation or individual slides without permission from Synopsys is prohibited.
Power Estimation – PVT Platform Example – 1\textsuperscript{st} order power model

Penalty when Cache miss ($\Delta_{\text{cache miss}}$)

Fixed power $f(V,f,\text{st})$

Fixed power $f(V,f,\text{st})$

Penalty for each bus transaction ($\Delta_{\text{transaction (length)}}$)

Penalty for each bus transaction ($\Delta_{\text{sys memory (length)}}$)

Penalty for each bus transaction ($\Delta_{\text{sys memory (length)}}$)

Penalty for each bus transaction ($\Delta_{\text{sys memory (length)}}$)

Fixed power $f(\text{freq},\text{St})$

* Under development
Dashboards Overview

- On-chip Power Reset Controller
- Power Dashboard
- Clock Dashboard
- Voltage Monitor
- SoC Voltage Dashboard
Power Analysis View Example
Part 2 : UPF Language in Design Flows

- UPF Target Design Styles
- UPF Conceptual model
- Synopsys UPF Flows
- Multi-Voltage Rule Checks
- Multi-Voltage Simulation Flow
- Logic Synthesis
- Design for Testing Flow
- Multi-Voltage Place & Route Flow
UPF Targets Design Styles using Advanced Power Management Techniques

- **Mainstream Techniques**

  - Clock Gating
  - Multi-Threshold

- **Advanced Techniques**
  - Multi-Voltage (MV)
  - MTCMOS power gating (shut down)
  - MV with power gating
  - Dynamic Voltage Frequency Scaling (DVFS)
UPF Conceptual Model

- Overlay power information on top of the design
Synopsys UPF Flow

Functional Verification

- VCS+MVSIM
- MVRC

Implementation

- RTL
- UPF

- Design Compiler
- Power Compiler

- Gate
- UPF

- IC Compiler

Equivalence Checking

- Ref
- Formality
- Impl

- PrimeTime
- PrimeTime PX

Timing/Power Signoff

- PrimeTime
- PrimeTime PX

- PrimeRail

This material is property of Synopsys Inc. Using this presentation or individual slides without permission from Synopsys is prohibited.
Multi-Voltage Rule Checking using MVRC

Implementation

- RTL
- UPF

Design Compiler

Power Compiler

Gate

UPF''

IC Compiler

Gate

UPF'''

PG Netlist

Static Functional Verification

- MVRC: RTL Checks

- MVRC: Netlist Checks

- MVRC: Final Signoff

This material is property of Synopsys Inc. Using this presentation or individual slides without permission from Synopsys is prohibited.
**MVRC**

**Structural Checks**

Domain 1
ON (1.2V) / OFF

Domain 2
0.9-1.2

PMIC/PMU

ISO-Enable

Domain 3
ON (1.2V)

Isolation

Level Shifters

Checks protection logic against:
- Missing cell
- Redundant cell
- Incorrect cell type
- Incorrect power domain
- Incorrect isolation polarity
- Incorrect iso-enable

<table>
<thead>
<tr>
<th>Mode</th>
<th>Domain1</th>
<th>Domain2</th>
<th>Domain3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mode1</td>
<td>1.2V</td>
<td>1.2V</td>
<td>1.2V</td>
</tr>
<tr>
<td>Mode2</td>
<td>Off</td>
<td>1.2V</td>
<td>1.2V</td>
</tr>
<tr>
<td>Mode3</td>
<td>Off</td>
<td>0.9V</td>
<td>1.2V</td>
</tr>
</tbody>
</table>

Structural checks performed using supplied MV state table

This material is property of Synopsys Inc. Using this presentation or individual slides without permission from Synopsys is prohibited.
**MVRC Architectural Checks**

- Architecture checks verify that isolation / sleep control signals are generated from the proper domain.

- ISO_Control is driven during Mode2, but in Mode3 it becomes HighZ.

<table>
<thead>
<tr>
<th></th>
<th>Domain 1</th>
<th>Domain 2</th>
<th>Domain 3</th>
<th>Domain 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mode1</td>
<td>1.2V</td>
<td>1.2V</td>
<td>1.2V</td>
<td>1.2V</td>
</tr>
<tr>
<td>Mode2</td>
<td>Off</td>
<td>1.2V</td>
<td>1.2V</td>
<td>1.2V</td>
</tr>
<tr>
<td>Mode3</td>
<td>Off</td>
<td>0.9V</td>
<td>[Red Circle]</td>
<td>1.2V</td>
</tr>
</tbody>
</table>
MVRC
Architectural Checks (2)

- Non-MV aware DFT and CTS tools can place buffers into an incorrect domain

- Structurally correct, but may lead to functional problems
  - Can be caught with test vectors, but MVRC can catch without vectors
MVSIM Flow

- UPF
- RTL/Netlist
- Testbench
- MVCMP
- MVDBGEN
- VCS + MVSIM
- Multi-Voltage VCD/FSDB
- APDB

This material is property of Synopsys Inc. Using this presentation or individual slides without permission from Synopsys is prohibited.
Multi-voltage Simulation Steps

> mvcmp -f simulation_file_list
> mvcmp -upf tb_ChipTop+.upf
> mvdbgen -top tb
> /usr/bin/gmake -f Makefile.ev all TOOL=vcs

- `simulation_file_list` consists of RTL verilog files and testbench files
- The testbench needs to be made power aware. This is done by creating a power domain for the testbench. Also, the power supply nets defined in the design need to be propagated up into the testbench, so that the testbench can control them. This is done with a testbench level UPF `- tb_ChipTop+.upf`
- `mvdbgen` is run in testbench module
Power-down and Power-up Sequence simulation Example

- GPRS register loaded
- Save asserted for retention
- ISO signal asserted
- GPRS output DATA clamped to 1
- GPRS shutdown signal asserted
- GPRS wake up
- Restore signal asserted
- Register values restored
Logic Synthesis

- `compile(_ultra)` recognizes multiple operating voltages and automatically synthesizes logic
- `compile(_ultra)` works with special cells
  - Maps isolation cells and retention registers based on directives
  - Hooks up control signals of retention registers
  - Performs always on synthesis
  - Optimizes level shifters
    - Automatic insertion of level shifter cells
    - Sizing of level shifters and isolation cells
    - Remapping from ISO/LS to ELS

- All multi-voltage features are available in Design Compiler Topographical mode as well
Always-on Synthesis

- Mark lib cells to be used as AO
- AO anchor pins (ISO, ELS, Switch, Ret control pins) are automatically inferred from PG-pin library
- Use `get_always_on_logic` to retrieve AO nets and cells
- To enable AO synthesis, set `enable_ao_synthesis` to `true`
- Compile/place_opt
Overview of DFT support in UPF mode

- All DFT MV features in non-UPF mode are available in UPF mode

- DFT MAX in UPF mode is easier to use vs. non-UPF mode
  - Automatic isolation and enable-level-shifter cell insertion as part of `insert_dft`
  - Correct handling of MV cell location
DFT flows supported in UPF mode

- **Standard scan**
  - Includes AutoFix, observe point insertion, user-defined test point insertion
  - Multiplexed flip-flop scan style only

- **Adaptive scan**
  - Default and High X-tolerance
  - Multiplexed flip-flop scan style only
Multi-Voltage-Aware DFT

- By default, scan stitching does not mix scan cells between power domains
- To allow mixing of scan-chains:

  - `set_scan_configuration -power_domain_mixing true`
  - `set_scan_configuration -voltage_mixing true`

![Diagram showing multi-voltage aware DFT with PD1, PD2, and PD top with voltages 0.7V, 0.9V, and 1.1V.]

- `voltage_mixing false`
- `power_domain_mixing false`

- `voltage_mixing true`
- `power_domain_mixing true`
IC Compiler UPF flow

- Read ddc from DC/DCT
- Read TDF IO/pin constraints
- Timing constraints are passed through ddc
- UPF power intent is passed through ddc
- Same UPF subset is supported in DC and ICC
- Most of special cells (LS, ISO, ELS, RR) are inserted in DC and maintained/optimized in ICC
Design Planning

Voltage Area Creation

- Create Voltage area for each power domain
- Rectilinear shapes are allowed
- Physically *nested voltage areas* supported

```
## CREATE VA FOR EACH PD
create_voltage_area \ 
  -power_domain MULT \ 
  -coord { 40 40 60 60 }
```
Design Planning

**PG Pin Hookup**

PN1 (switch input)

PN2 (switch output, primary)

- PG pins of standard cells hooked up to PG nets automatically
- PG connections are derived from
  - PG pin syntax in the library
  - Scoped and mapped power net UPF objects
- Tie-off nets are also hooked up to correct PG net
- Checks for any PG connection violations

```python
derive_pg_connection
check_mv_design -power_nets
```

connect_pg_nets is recommended for physical only cells

This material is property of Synopsys Inc. Using this presentation or individual slides without permission from Synopsys is prohibited.
Switch Cell Placement

```
add_header_footer_cell_array
-lib_cell "mult_sw"
-voltage_area MULT
-design Multiplier
-x_increment 63
-y_increment 8
```
Switch Cell Power Planning
Automated Power Switch Handling
Rapid Analysis of Power Network Configuration

Problem:  *How many power switches? Where do you put them?*
Too few switches cause **IR drop** issues
Too many switches take valuable chip area

<table>
<thead>
<tr>
<th></th>
<th>MAX VD(mV)</th>
<th>Area(%)</th>
<th>Res(Ω)</th>
<th>Total N</th>
<th>X pitch</th>
<th>Y pitch</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>284.092</td>
<td>9.03</td>
<td>10</td>
<td>338</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>B</td>
<td>280.707</td>
<td>6.09</td>
<td>10</td>
<td>228</td>
<td>40</td>
<td>25</td>
</tr>
<tr>
<td>C</td>
<td>259.819</td>
<td>8.02</td>
<td>10</td>
<td>300</td>
<td>40</td>
<td>20</td>
</tr>
</tbody>
</table>

Automatically generate various options...

Designer chooses best option

Implementation automatically optimized

This material is property of Synopsys Inc. Using this presentation or individual slides without permission from Synopsys is prohibited
By using MCCM for a MV design,
- “Dynamic voltage scaling” designs can be implemented
- i.e. the design блокs can operate at different voltages at different

\begin{itemize}
  \item \textbf{S0:} Design @ HV, RAMs @ LV
  \item \textbf{S1:} Top, Mult @ HV
  \item GPRS, RAMs @ LV
\end{itemize}
MV Aware Placement

- MV aware placement and optimization
- Multi-site row support
- Special Level Shifter and Isolation Cells handling
- High Fanout Net Synthesis (HFNS)
- Routing estimation detours around VA

This material is property of Synopsys Inc. Using this presentation or individual slides without permission from Synopsys is prohibited.
MV Aware Clock Tree Synthesis

- Register clusters are created respecting voltage areas
- Clock routing is confined to voltage area
- Tracing through level shifters and enable level shifters
Part 3: UPF Language Syntax

- create_power_domain
- create_supply_port
- create_supply_net
- connect_supply_net
- set_domain_supply_net
- connect_supply_net
- add_port_state
- create_pst
- add_pst_state
- create_power_switch
- set_retention
- set_retention_control
- map_retention
- set_isolation
- set_isolation_control
- set_level_shifter
create_power_domain

create_power_domain <domain_name>
  [-elements <list of hierarchical instances>]
  [-scope <instance_name>]
  
  - Creates a power domain at the current scope, level of hierarchy
  - **-elements** to define specific instances to include in power domain
  - **-scope** to create the power domain in another scope, level of hierarchy
    - To define a power domain for a child design at the child’s level of hierarchy
Scope of a Power Domain

- Power domain created at top level scope
  ```
  create_power_domain Top
  create_power_domain GPRS -elements GPRs
  ```
- Power domain created as GPRS
- Subsequently all UPF components for power domain GPRS will be created at top level
  ```
  create_supply_port VDDG -domain GPRS
  ```
Scope of a Power Domain

- Power domain created at lower level scope
  ```
  create_power_domain
  GPRS -elements GPRs
  -scope GPRs
  ```

- Power domain created as `GPRs/GPRS`

- Subsequently all UPF components for power domain GPRS will be created at GPRs level
  ```
  create_supply_port
  VDDG -domain GPRS
  ```
Supply Network

- UPF can specify a complete power supply network
  - Supply ports
  - Supply nets
  - Power switches
  - Implicit connections based on PD
  - Explicit supply connections

- UPF commands
  - `create_supply_ports`
  - `create_supply_nets`
  - `create_power_switch`
  - `connect_supply_net`
  - `set_domain_supply_net`
create_supply_port <port_name>

[-domain <domain name>]

- port_name needs to be unique at the level of hierarchy it is defined
- -domain is used to specify supply ports inside another power domain
- For designs with multiple power domains at the same scope/level of hierarchy, supply ports are available to all power domains defined at that scope

>create_supply_port VDD -domain child
create_supply_net

create_supply_net <supply_net_name>
- domain <domain_name>
[-reuse]

- supply_net_name must be a unique identifier
- -domain specifies the power domain in which the supply net is to be created
- -reuse specifies that the listed supply net name is to be re-used as a supply net inside the power domain specified by the -domain option
Re-use of supply nets

➢ To make supply net VDD available to PD2, the -reuse option needs to be used when defining the supply net VDD in PD2
>
create_power_domain PD1 -elements {A B}
create_power_domain PD2 -elements {C D}
create_supply_net VDD -domain PD1
create_supply_net VDD -domain PD2 -reuse

➢ Supply net VDD is now available in both PD1 and PD2
connect_supply_net

connect_supply_net <supply_net> [-ports list]

- To connect a supply net to a supply port at current level of hierarchy, self
  
  ```
  connect_supply_net VDD -ports VDD
  ```

- To connect a supply net to a child’s supply port
  
  ```
  connect_supply_net VDD -ports child/VDD
  ```

- To connect a supply net to both self and child
  
  ```
  connect_supply_net VDD -ports {VDD child/VDD}
  ```
set_domain_supply_net

```
set_domain_supply_net <domain_name> \
    -primary_supply_net <power_supply_net> \
    -primary_ground_net <ground_supply_net>
```

- Tells tools what are the default power and ground connections for cells in a power domain
Dynamic Voltage and Frequency Scaling (DVFS): Power State Tables

- Capture dynamic voltage scaling (DVS/DVFS) and shutdown scenarios with Power State Table (PST)
- UPF Commands
  - `add_port_state`
  - `create_pst`
  - `add_pst_state`

<table>
<thead>
<tr>
<th>Power State</th>
<th>Vdd1</th>
<th>Vdd2</th>
<th>Vdd3</th>
</tr>
</thead>
<tbody>
<tr>
<td>PwrState1</td>
<td>0.8V</td>
<td>0.8V</td>
<td>0.8V</td>
</tr>
<tr>
<td>PwrState2</td>
<td>0.8V</td>
<td>0.9V</td>
<td>off</td>
</tr>
</tbody>
</table>
Power State Table

- PST defines all possible power states for a design
- Should be defined at the top level of a design
- Only one PST can be defined for a design

```<add_port_state> <port_name> {<state> {name <nom>|<min nom max>|<off>}}}
  - Defines all possible state information to a supply port
```

```<create_pst> <table_name> -supplies {list}>
  - Creates a PST using a specific order of supply nets
```

```<add_pst_state> <state_name> -pst <table_name> -state <supply_states>
  - Defines states of each of the supply nets for one possible state of the design
```
create_power_switch

create_power_switch <switch_name>
-domain <domain_name>
-output_supply_port <port_name
 supply_net_name>
{ -input_supply_port <port_name
 supply_net_name> }
{ -control_port <port_name net_name> }
[-ack_port <port_name net_name>]

- Multiple control and acknowledge signals may be given to
  the power switch defined
- ICC will expand this into a power switch network
Create power switch for GPRS

```plaintext
create_power_switch gprs_sw
  -domain GPRS
  -input_supply_port {in VDDG} -output_supply_port {out VDDGS} -control_port {sleep PwrCtrl/sleep} -on_state {state2002 in {!sleep}}
```
Isolation Cells

\texttt{set\_isolation}\ <\texttt{isolation\_strategy}>\n- \texttt{domain power\_domain} \n- \texttt{isolation\_power\_net} <\texttt{isolation\_power\_net}> \n- \texttt{isolation\_ground\_net} <\texttt{isolation\_ground\_net}> \n  [\texttt{-clamp\_value 0 | 1 | latch}] \n  [\texttt{-applies\_to inputs | outputs | both}] \n  [\texttt{-elements objects}] \n  [\texttt{-no\_isolation}]

➢ The power net given to \texttt{isolation\_power\_net} and \texttt{isolation\_ground\_net} needs to be more always_on than the domain’s primary power or ground net
Isolation Cells

**set** _isolation_control_

```plaintext
set_isolation_control <isolation_strategy>
  -domain power_domain
  -isolation_signal <isolation_signal>
  [-isolation_sense 0 | 1]
  [-location self | parent]
```

- The `isolation_signal` needs to exist in the scope in which the isolation strategy is defined
Isolation Strategy

- Isolation strategy for GPRS

```plaintext
set_isolation
gprs_iso_out -domain GPRS
-isolation_power_net VDD
-isolation_ground_net VSS
-clamp_value 1
-applies_to outputs

set_isolation_control
gprs_iso_out -domain GPRS
-isolation_signal
PwrCtrl/isolate_ctrl
-isolation_sense low
-location parent
```
set_retention

```plaintext
set_retention <retention_strategy> -domain <power_domain> -retention_power_net <retention_power_net> -retention_ground_net <retention_ground_net> [-elements objects]
```

- The power net used must be more always on that the power domain’s primary power
- Unless elements are specified, this constraint will apply to all registers in the power domain
set_retention_control

set_retention_control <retention_strategy>
-domain power_domain
-save_signal {{net_name <high | low >}}
-restore_signal {{net_name <high | low >}}

➢ The save and restore signals will be stitched up to the scope the set_retention_control command was defined at
  ▪ If set_retention_control was defined at the top scope, save and restore will be stitched up to the top of the design
map_retention_cell

map_retention_cell <retention_strategy>
-domain power_domain
[-lib_cells lib_cells]
[-lib_cell_type lib_cell_type]
[-elements objects]

- Directs DC to map the specified sequential cells to a specific retention cell during compile(_ultra)
Retention Register Strategy

- Retention strategy for GPRS

```plaintext
set_retention gprs_ret -domain GPRS
-retention_power_net VDDG
-retention_ground_net VSS

set_retention_control gprs_ret -domain GPRS
-save_signal {PwrCtrl/retn high} -restore_signal {PwrCtrl/restore high}

map_retention_cell gprs_ret -domain GPRS
-lib_cell_type RSDFCD1
```
Level Shifters

set_level_shifter <level_shifter_name>
-domain <domain_name>
[-elements list]
[-applies_to <inputs | outputs | both>]
[-threshold value]
[-rule <low_to_high | high_to_low | both>]
[-location <self | parent | fanout | automatic>]
[-no_shift]

The default for level shifters is ‘-rule both’ and ‘-location automatic’

‘-location automatic’ allows the tool is determine what is the best location for level shifter insertion
Predictable Success

Thank you for your attention!