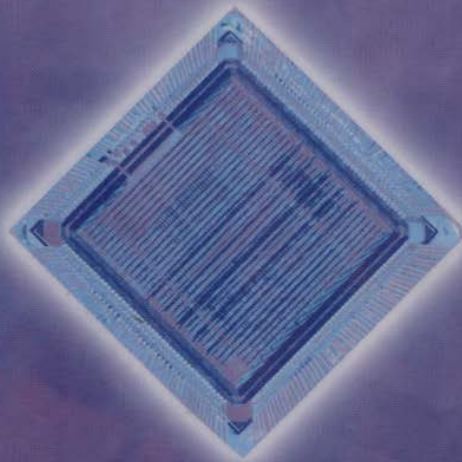


# Low Power Methodology Manual

## For System-on-Chip Design



Michael Keating      David Flynn  
Robert Aitken      Alan Gibbons      Kaijian Shi

 Springer

U.C. BERKELEY LIBRARY

---

Michael Keating • David Flynn •  
Robert Aitken • Alan Gibbons • Kaijian Shi

# Low Power Methodology Manual

For System-on-Chip Design

 Springer

engi  
015783169

Michael Keating  
Synopsis, Inc.  
Palo Alto, CA  
USA

David Flynn  
ARM Limited  
Cambridge  
United Kingdom

Robert Aitken  
ARM, Inc.  
Almaden, CA  
USA

Alan Gibbons  
Synopsis, Inc.  
Northampton  
United Kingdom

Kaijian Shi  
Synopsis, Inc.  
Dallas, TX  
USA

Library of Congress Control Number: 2007928355

ISBN 978-0-387-71818-7

e-ISBN 978-0-387-71819-4

Printed on acid-free paper.

Copyright © 2007 by Synopsis, Inc. & ARM Limited. All rights reserved.

All rights reserved. This work may not be translated or copied in whole or in part without the written permission of the publisher (Springer Science+Business Media, LLC, 233 Spring Street, New York, NY 10013, USA), except for brief excerpts in connection with reviews or scholarly analysis. Use in connection with any form of information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed is forbidden. The use in this publication of trade names, trademarks, service marks and similar terms, even if they are not identified as such, is not to be taken as an expression of opinion as to whether or not they are subject to proprietary rights.

9 8 7 6 5 4 3 2 1

springer.com

cmr 11/27/07

TK7874  
.66  
L67  
2007  
ENGI

## TRADEMARKS

Synopsys and NanoSim are registered trademarks of Synopsys, Inc.

ARM and AMBA are registered trademarks of ARM Limited. ARM926EJ-S, ARM1176JZF-S, AHB and APB are trademarks of ARM Limited. Artisan and Artisan Components are registered trademarks of ARM Physical IP, Inc.

“ARM” is used to represent ARM Holdings plc; its operating company ARM Limited; and the regional subsidiaries ARM INC.; ARM KK; ARM Korea Ltd.; ARM Taiwan; ARM France SAS; ARM Consulting (Shanghai) Co. Ltd.; ARM Belgium N.V.; AXYS Design Automation Inc.; AXYS GmbH; ARM Embedded Technologies Pvt. Ltd.; and ARM, Inc. and ARM Norway, AS.

All other brands or product names are the property of their respective holders.

## DISCLAIMER

All content included in this Low Power Methodology Manual is the result of the combined efforts of ARM Limited and Synopsys, Inc. Because of the possibility of human or mechanical error, neither the authors, ARM Limited, Synopsys, Inc., nor any of their affiliates, including but not limited to Springer Science+Business Media, LLC, guarantees the accuracy, adequacy or completeness of any information contained herein and are not responsible for any errors or omissions, or for the results obtained from the use of such information. THERE ARE NO EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE relating to the Low Power Methodology Manual. In no event shall the authors, ARM Limited, Synopsys, Inc., or their affiliates be liable for any indirect, special or consequential damages in connection with the information provided herein.

# *Table of Contents*

|                                                                  |           |
|------------------------------------------------------------------|-----------|
| <b>Preface.....</b>                                              | <b>XV</b> |
| <b>1 Introduction.....</b>                                       | <b>1</b>  |
| 1.1 Overview .....                                               | 1         |
| 1.2 Scope of the Problem.....                                    | 2         |
| 1.3 Power vs. Energy .....                                       | 3         |
| 1.4 Dynamic Power .....                                          | 4         |
| 1.5 The Conflict Between Dynamic and Static Power .....          | 7         |
| 1.6 Static Power.....                                            | 8         |
| 1.7 Purpose of This Book .....                                   | 10        |
| <b>2 Standard Low Power Methods .....</b>                        | <b>13</b> |
| 2.1 Clock Gating.....                                            | 13        |
| 2.2 Gate Level Power Optimization .....                          | 15        |
| 2.3 Multi VDD.....                                               | 16        |
| 2.4 Multi-Threshold Logic .....                                  | 17        |
| 2.5 Summary of the Impact of Standard Low Power Techniques ..... | 19        |
| <b>3 Multi-Voltage Design .....</b>                              | <b>21</b> |
| 3.1 Challenges in Multi-Voltage Designs .....                    | 22        |
| 3.2 Voltage Scaling Interfaces – Level Shifters.....             | 22        |
| 3.2.1 Unidirectional Level Shifters .....                        | 23        |

|          |                                                                 |           |
|----------|-----------------------------------------------------------------|-----------|
| 3.2.2    | Level Shifters – High to Low Voltage Translation.....           | 23        |
| 3.2.3    | Level Shifters – Low-to-High Voltage.....                       | 24        |
| 3.2.4    | Level Shifter Placement .....                                   | 25        |
| 3.2.5    | Automation and Level Shifters.....                              | 27        |
| 3.2.6    | Level Shifter Recommendations and Pitfalls .....                | 28        |
| 3.3      | Timing Issues in Multi-Voltage Designs .....                    | 29        |
| 3.3.1    | Clocks .....                                                    | 29        |
| 3.3.2    | Static Timing Analysis .....                                    | 30        |
| 3.4      | Power Planning for Multi-Voltage Design .....                   | 30        |
| 3.5      | System Design Issues with Multi-Voltage Designs.....            | 31        |
| <b>4</b> | <b>Power Gating Overview .....</b>                              | <b>33</b> |
| 4.1      | Dynamic and Leakage power profiles.....                         | 33        |
| 4.2      | Impact of Power Gating on Classes of Sub-systems.....           | 36        |
| 4.3      | Principles of Power Gating Design .....                         | 37        |
| 4.3.1    | Power Switching – Fine Grain vs. Coarse Grain.....              | 38        |
| 4.3.2    | The Challenges of Power Gating.....                             | 39        |
| <b>5</b> | <b>Designing Power Gating.....</b>                              | <b>41</b> |
| 5.1      | Switching Fabric Design .....                                   | 42        |
| 5.1.1    | Controlling the Switching Fabric .....                          | 44        |
| 5.1.2    | Recommendations and Pitfalls for Power Gating Control .....     | 44        |
| 5.2      | Signal Isolation .....                                          | 45        |
| 5.2.1    | Signal Isolation techniques.....                                | 45        |
| 5.2.2    | Output or Input Isolation .....                                 | 47        |
| 5.2.3    | Interface Protocols and Isolation .....                         | 48        |
| 5.2.4    | Recommendations and Pitfalls for Isolation.....                 | 50        |
| 5.3      | State Retention and Restoration Methods .....                   | 50        |
| 5.3.1    | State Retention Using Scan Chains .....                         | 51        |
| 5.3.2    | Retention Registers.....                                        | 54        |
| 5.3.3    | Power Controller Design for Retention.....                      | 56        |
| 5.3.4    | Partial vs. Full State Retention .....                          | 56        |
| 5.3.5    | System Level Issues and Retention .....                         | 58        |
| 5.3.6    | Recommendations and Pitfalls for State Retention .....          | 58        |
| 5.4      | Power Gating Control.....                                       | 59        |
| 5.4.1    | Power Control Sequencing.....                                   | 60        |
| 5.4.2    | Handshake Protocols .....                                       | 61        |
| 5.4.3    | Recommendations and Pitfalls for Power Gating Controllers ..... | 63        |
| 5.5      | Power Gating Design Verification – RTL Simulation.....          | 63        |

|          |                                                            |            |
|----------|------------------------------------------------------------|------------|
| 5.5.1    | Inferring Power Gating Behavior in RTL.....                | 64         |
| 5.5.2    | Inferring Power Gating and Retention Behavior in RTL ..... | 68         |
| 5.6      | Design For Test considerations .....                       | 70         |
| 5.6.1    | Power Gating Controls .....                                | 70         |
| 5.6.2    | Power Limitations during Scan Test.....                    | 71         |
| 5.6.3    | Testing the Switching Network .....                        | 71         |
| 5.6.4    | Testing Isolation and Retention .....                      | 72         |
| 5.6.5    | Testing the Power Gating Controller .....                  | 73         |
| <b>6</b> | <b>Architectural Issues for Power Gating .....</b>         | <b>75</b>  |
| 6.1      | Hierarchy and Power Gating .....                           | 75         |
| 6.2      | Power Networks and Their Control.....                      | 78         |
| 6.2.1    | External Power Rail Switching.....                         | 79         |
| 6.2.2    | On-chip Power Gating .....                                 | 81         |
| 6.3      | Power State Tables and Always On Regions.....              | 82         |
| <b>7</b> | <b>A Power Gating Example.....</b>                         | <b>85</b>  |
| 7.1      | Leakage Modes Supported .....                              | 85         |
| 7.2      | Design partitioning .....                                  | 88         |
| 7.3      | Isolation .....                                            | 92         |
| 7.4      | Retention.....                                             | 94         |
| 7.5      | Inferring Power Gating and Retention .....                 | 95         |
| 7.6      | Measurements and Analysis .....                            | 96         |
| <b>8</b> | <b>IP Design for Low Power .....</b>                       | <b>101</b> |
| 8.1      | Architecture and Partitioning for Power Gating.....        | 102        |
| 8.1.1    | How and When to Shut Down.....                             | 103        |
| 8.1.2    | What to Shut Down and What to Keep Alive .....             | 103        |
| 8.2      | Power Controller Design for the USB OTG.....               | 105        |
| 8.3      | Issues in Designing Portable Power Controllers .....       | 108        |
| 8.4      | Clocks and Resets .....                                    | 109        |
| 8.5      | Verification .....                                         | 109        |
| 8.6      | Packaging IP for Reuse with Power Intent.....              | 110        |
| 8.7      | UPF for the USB OTG Core .....                             | 111        |
| 8.8      | USB OTG Power Gating Controller State Machine.....         | 114        |

|           |                                                                        |            |
|-----------|------------------------------------------------------------------------|------------|
| <b>9</b>  | <b>Frequency and Voltage Scaling Design .....</b>                      | <b>121</b> |
| 9.1       | Dynamic Power and Energy .....                                         | 122        |
| 9.2       | Voltage Scaling Approaches .....                                       | 125        |
| 9.3       | Dynamic Voltage and Frequency Scaling (DVFS) .....                     | 125        |
| 9.4       | CPU Subsystem Design Issues .....                                      | 129        |
| 9.5       | Adaptive Voltage Scaling (AVS) .....                                   | 130        |
| 9.6       | Level Shifters and Isolation .....                                     | 131        |
| 9.7       | Voltage Scaling Interfaces – Effect on Synchronous Timing .....        | 132        |
| 9.8       | Control of Voltage Scaling .....                                       | 136        |
| <b>10</b> | <b>Examples of Voltage and Frequency Scaling Design .....</b>          | <b>139</b> |
| 10.1      | Voltage Scaling - A Worked Example for UMC 130nm .....                 | 139        |
| 10.1.1    | ULTRA926 System Design Block Diagram .....                             | 140        |
| 10.1.2    | Voltage/Frequency Range Exploration .....                              | 141        |
| 10.1.3    | Synchronous Design Constraints .....                                   | 144        |
| 10.1.4    | Simulated (predicted) Energy Savings Analysis .....                    | 145        |
| 10.1.5    | Silicon-Measured Power and Performance Analysis .....                  | 145        |
| 10.1.6    | Silicon-Measured ULTRA926 DVFS Energy Savings Analysis ..              | 147        |
| 10.2      | Voltage Scaling – A worked Example for TSMC 65nm .....                 | 150        |
| 10.2.1    | ATLAS926 Case Study .....                                              | 150        |
| 10.2.2    | Voltage/Frequency Range Exploration .....                              | 151        |
| 10.2.3    | Silicon-Measured Power and Performance Analysis .....                  | 151        |
| <b>11</b> | <b>Implementing Multi-Voltage, Power Gated Designs .....</b>           | <b>155</b> |
| 11.1      | Design Partitioning .....                                              | 158        |
| 11.1.1    | Logical and Physical Hierarchy .....                                   | 158        |
| 11.1.2    | Critical Path Timing .....                                             | 160        |
| 11.2      | Design Flow Overview .....                                             | 160        |
| 11.3      | Synthesis .....                                                        | 162        |
| 11.3.1    | Power Intent .....                                                     | 162        |
| 11.3.2    | Defining Power Domains and Power Connectivity .....                    | 162        |
| 11.3.3    | Isolation Cell Insertion .....                                         | 163        |
| 11.3.4    | Retention Register Insertion .....                                     | 164        |
| 11.3.5    | Level Shifter Insertion .....                                          | 166        |
| 11.3.6    | Scan Synthesis .....                                                   | 168        |
| 11.3.7    | Always-On Network Synthesis .....                                      | 170        |
| 11.4      | Multi Corner Multi Mode Optimization with Voltage Scaling Designs .... | 171        |
| 11.5      | Design Planning .....                                                  | 173        |

|           |                                                          |            |
|-----------|----------------------------------------------------------|------------|
| 11.5.1    | Creating Voltage Areas .....                             | 173        |
| 11.5.2    | Power Gating Topologies .....                            | 175        |
| 11.5.3    | In-rush Current Management .....                         | 176        |
| 11.5.4    | Recommendations: .....                                   | 176        |
| 11.6      | Power Planning.....                                      | 177        |
| 11.6.1    | Decoupling Capacitor Insertion.....                      | 179        |
| 11.7      | Clock Tree Synthesis .....                               | 180        |
| 11.8      | Power Analysis .....                                     | 183        |
| 11.9      | Timing Analysis .....                                    | 184        |
| 11.10     | Low Power Validation.....                                | 185        |
| 11.11     | Manufacturing Test.....                                  | 185        |
| <b>12</b> | <b>Physical Libraries .....</b>                          | <b>187</b> |
| 12.1      | Standard Cell Libraries.....                             | 187        |
| 12.1.1    | Modeling of Standard Cell Libraries .....                | 188        |
| 12.1.2    | Characterization of Standard Cell Libraries .....        | 189        |
| 12.2      | Special Cells - Isolation Cells.....                     | 190        |
| 12.2.1    | Signal Isolation .....                                   | 191        |
| 12.2.2    | Output Isolation vs. Input Isolation .....               | 193        |
| 12.2.3    | Sneak DC Leakage Paths .....                             | 193        |
| 12.2.4    | Recommendations .....                                    | 194        |
| 12.3      | Special Cells - Level Shifters .....                     | 195        |
| 12.4      | Memories.....                                            | 198        |
| 12.4.1    | RAMs for Multi-Voltage Power Gated Designs.....          | 199        |
| 12.4.2    | Memories and Retention.....                              | 200        |
| 12.5      | Power Gating Strategies and Structures .....             | 200        |
| 12.5.1    | Power Gating Structures.....                             | 201        |
| 12.5.2    | Recommendations – Coarse Grain vs. Fine Grain .....      | 204        |
| 12.6      | Power Gating Cells.....                                  | 204        |
| 12.7      | Power Gated Standard Cell Libraries.....                 | 206        |
| <b>13</b> | <b>Retention Register Design .....</b>                   | <b>209</b> |
| 13.1      | Retention Registers.....                                 | 209        |
| 13.1.1    | Single Pin “Live Slave” Retention Registers .....        | 209        |
| 13.1.2    | Dual Control Signal “Balloon” Retention Register .....   | 212        |
| 13.1.3    | Single Control Signal “Balloon” Retention Register ..... | 215        |
| 13.1.4    | Retention Register: Relative layout.....                 | 218        |
| 13.2      | Memory Retention Methods.....                            | 219        |
| 13.2.1    | VDD Retention Method .....                               | 219        |

|                   |                                                                       |            |
|-------------------|-----------------------------------------------------------------------|------------|
| 13.2.2            | Source-diode Biasing Method .....                                     | 219        |
| 13.2.3            | Source Biasing Method .....                                           | 221        |
| 13.2.4            | Retention Latency Reduction Methods .....                             | 222        |
| <b>14</b>         | <b>Design of the Power Switching Network.....</b>                     | <b>225</b> |
| 14.1              | Ring vs. Grid Style .....                                             | 225        |
| 14.1.1            | Ring Style Implementation.....                                        | 226        |
| 14.1.2            | Grid Style Implementation .....                                       | 227        |
| 14.1.3            | Row and Column Grids .....                                            | 229        |
| 14.1.4            | Hybrid Style Implementation .....                                     | 231        |
| 14.1.5            | Recommendations - Ring vs. Grid Style .....                           | 231        |
| 14.2              | Header vs. Footer Switch .....                                        | 232        |
| 14.2.1            | Switch Efficiency Considerations.....                                 | 232        |
| 14.2.2            | Area Efficiency Consideration and L/W Choice .....                    | 234        |
| 14.2.3            | Body Bias Considerations.....                                         | 235        |
| 14.2.4            | System Level Design Consideration.....                                | 235        |
| 14.2.5            | Recommendations – Header vs. Footer.....                              | 235        |
| 14.3              | Rail vs. Strap VDD Supply .....                                       | 236        |
| 14.3.1            | Parallel Rail VDD Distribution .....                                  | 236        |
| 14.3.2            | Power Strap VDD Distribution .....                                    | 238        |
| 14.3.3            | Recommendations for Supply Distribution .....                         | 239        |
| 14.4              | A Sleep Transistor Example.....                                       | 239        |
| 14.5              | Wakeup Current and Latency Control Methods.....                       | 240        |
| 14.5.1            | Single Daisy chain sleep transistor distribution .....                | 241        |
| 14.5.2            | Dual Daisy chain sleep transistor distribution.....                   | 241        |
| 14.5.3            | Parallel Short Chain Distribution of the Main Sleep Transistor ...    | 243        |
| 14.5.4            | Main Chain Turn-on Control .....                                      | 243        |
| 14.5.5            | Buffer Delay Based Main Chain Turn-on Control .....                   | 243        |
| 14.5.6            | Programmable Main Chain Turn-on Control.....                          | 244        |
| 14.5.7            | Power-off Latency Reduction .....                                     | 244        |
| 14.5.8            | Recommendations for Power Switching Control.....                      | 245        |
| 14.6              | An Example of a Dual Daisy Chain Sleep Transistor Implementation..... | 246        |
| <b>APPENDIX A</b> | <b>Sleep Transistor Design.....</b>                                   | <b>249</b> |
| A.1               | Sleep Transistor Design Metrics .....                                 | 250        |
| A.1.1             | Switch Efficiency .....                                               | 250        |
| A.1.2             | Area Efficiency .....                                                 | 253        |
| A.1.3             | IR Drop.....                                                          | 253        |
| A.1.4             | Normal vs. Reverse Body Bias .....                                    | 254        |

|                                                |            |
|------------------------------------------------|------------|
| A.1.5 Recommendations .....                    | 259        |
| A.2 Layout Design for Area Efficiency .....    | 260        |
| A.2.1 Recommendations .....                    | 262        |
| A.3 Single Row vs. Double Row .....            | 262        |
| A.3.1 Recommendations .....                    | 263        |
| A.4 In-rush Current and Latency Analysis ..... | 263        |
| <b>APPENDIX B UPF Command Syntax.....</b>      | <b>267</b> |
| B.1 add_pst_state .....                        | 268        |
| B.2 connect_supply_net .....                   | 269        |
| B.3 create_power_domain .....                  | 271        |
| B.4 create_power_switch .....                  | 273        |
| B.5 create_pst .....                           | 275        |
| B.6 create_supply_net .....                    | 276        |
| B.7 create_supply_port .....                   | 277        |
| B.8 set_domain_supply_net .....                | 278        |
| B.9 set_isolation .....                        | 279        |
| B.10 set_isolation_control .....               | 281        |
| B.11 set_level_shifter .....                   | 283        |
| B.12 set_retention .....                       | 285        |
| B.13 set_retention_control .....               | 287        |
| B.14 set_scope .....                           | 288        |
| <b>Glossary .....</b>                          | <b>291</b> |
| <b>Bibliography .....</b>                      | <b>293</b> |
| <b>Index .....</b>                             | <b>297</b> |

# *Preface*

The Low Power Methodology Manual is the outcome of a decade-long collaboration between ARM and Synopsys commercially and the two of us personally. In 1997 ARM and Synopsys worked together to develop a synthesizable ARM7 core. Dave was the ARM lead on the project; Mike's team executed the Synopsys side of the project. This led to a similar project on the ARM9.

Shortly after these projects, the two of us embarked on a series of technology demonstration projects. We both felt that we needed to use our products as our customers do in order to understand how to make these products better. So we developed a test chip that combined ARM and Synopsys IP and took it through to silicon. We did the RTL design and verification personally, and borrowed resources to do the implementation. The experience was incredibly illuminating, and we hope it contributed to improving the IP and tools from both companies.

We quickly realized that low power was one of the key concerns of our customers, and SoC designers in general. So we followed our initial project with several low power technology demonstration projects. The final project was the SALT (Synopsys ARM Low-power Technology demonstrator) project, for which we received working silicon late last year. These projects explored clock gating, multi-voltage, dynamic voltage scaling, and power gating. In all these projects we found that there is no substitute for direct first-hand experience doing low-power IP-based designs. We learned, in the most concrete way possible, exactly what our customers go through on an SoC design.

For years we have been talking about writing a book on low power design. With our experience on the SALT project, our work with customers on low power designs, and

our participation in developing the UPF low-power standard, we feel that we are finally in a position to publish our insights and perspectives.

In doing so, we have enlisted the aid of our co-authors. The two of us are primarily front-end engineers, with a background in system architecture and RTL design. Kaijian and Rob bring a great depth of technical expertise in the physical and circuit design aspects of low power. Alan has developed low power flows for the ARM processors and did the implementation of SALT. As a result, he brings a unique perspective on the implementation issues in low power design.

We cannot overstate the contribution of our co-authors. Without their insights and expertise - as well as the material they contributed directly - this book could not have been written.

Like all our joint projects, this book was partly a formal joint project of the two companies and partly (perhaps mostly) driven by the personal commitment of the authors, aided and abetted by many others. We got considerable help from many people for whom this was not part of their job description. These kind souls took time out of their busy schedules, including evenings and weekends, to help us at every step of our journey, from the first joint chip development to the completion of this book. They helped in the architecture, design and tape out of test chips, the building and debugging of boards, and the review and editing of the final manuscript.

It is impossible to list them all, but we list some of the many who contributed to this effort: Anwar Awad, John Biggs, Pin-Hung Chen, Sachin Rai, David Howard, and Sachin Idgunji.

We would also like to thank the staffs of TSMC and UMC for fabricating the technology demonstrators and enabling us to derive the results referenced in the worked examples.

Dave Flynn  
Cambridge, UK

Mike Keating  
Palo Alto, CA

# *Standard Low Power Methods*

There are a number of power reduction methods that have been used for some time, and which are mature technologies. This chapter describes some of these approaches to low power design.:

- Clock Gating
- Gate Level Power Optimization
- Multi- $V_{DD}$
- Multi- $V_T$

## **2.1 Clock Gating**

A significant fraction of the dynamic power in a chip is in the distribution network of the clock. Up to 50% or even more of the dynamic power can be spent in the clock buffers. This result makes intuitive sense since these buffers have the highest toggle rate in the system, there are lots of them, and they often have a high drive strength to minimize clock delay. In addition, the flops receiving the clock dissipate some dynamic power even if the input and output remain the same.

The most common way to reduce this power is to turn clocks off when they are not required. This approach is known as clock gating.

Modern design tools support automatic clock gating: they can identify circuits where clock gating can be inserted without changing the function of the logic. Figure 2-1 shows how this works.

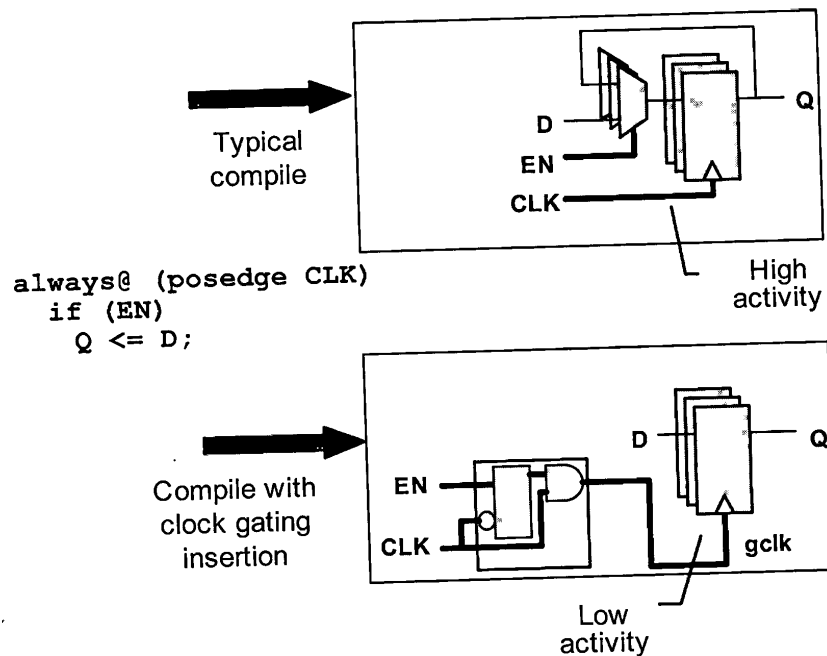


Figure 2-1 Clock Gating

In the original RTL, the register is updated or not depending on a variable (EN). The same result can be achieved by gating the clock based on the same variable.

If the registers involved are single bits, then a small savings occurs. If they are, say, 32 bit registers, then one clock gating cell can gate the clock to all 32 registers (and any buffers in their clock trees). This can result in considerable power savings.

In the early days of RTL design, engineers would code clock gating circuits explicitly in the RTL. This approach is error prone – it is very easy to create a clock gating circuit that glitches during gating, producing functional errors. Today, most libraries include specific clock gating cells that are recognized by the synthesis tool. The combination of explicit clock gating cells and automatic insertion makes clock gating a simple and reliable way of reducing power. No change to the RTL is required to implement this style of clock gating.

### Results

In a recent paper [1], Pokhrel reports on a unique opportunity his team recently had to compare a (nearly) identical chip implemented both with and without clock gating. As a power reduction project, an existing 180nm chip without clock gating was re-

implemented in the same technology with clock gating. Only minor changes in the logic were implemented (some small blocks were removed and replaced by other blocks, for a small net increase in functionality).

Pokhrel reports an area reduction of 20% and a power savings of 34% to 43%, depending on the operating mode. (This savings was realized on the clock gated part of the chip; the processor was a hard macro and not clock gated. Power measurements were made on the whole chip when the processor was in IDLE mode; that is, the processor was turned off.) The power measurements are from actual silicon.

The area savings is due to the fact that a single clock gating cell takes the place of multiple muxes.

Pokhrel makes a couple of interesting observations:

- After some analysis and experiments, the team decided to use clock gating only on registers with a bit-width of at least three. They found that clock gating on one-bit registers was not power or area efficient.
- Much of the power savings was due to the fact that the clock gating cells were placed early in the clock path. Approximately 60% of the clock buffers came after the clock gating cell, and so had their activity reduce to zero during gating.

## 2.2 Gate Level Power Optimization

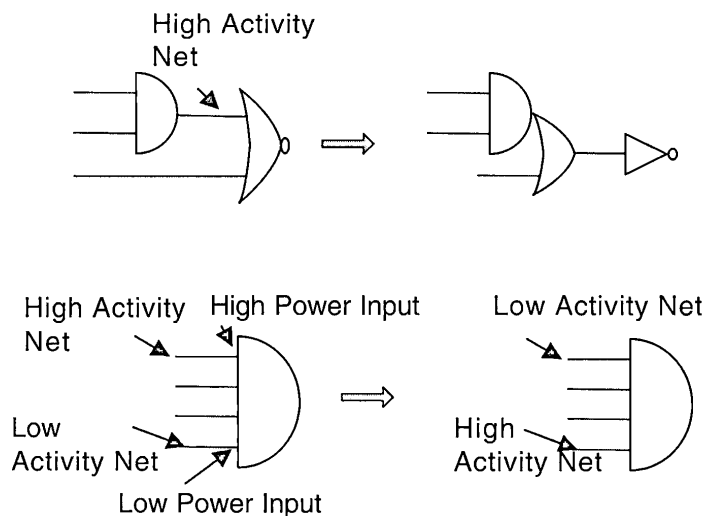


**Figure 2-2** Examples of Gate Level Optimizations

In addition to clock gating, there are a number of logic optimizations that the tools can perform to minimize dynamic power. Figure 2-2 shows two of these optimizations.

At the top of the figure, an AND gate output has a particularly high activity. Because it is followed by a NOR gate, it is possible to re-map the two gates to an AND-OR gate plus an inverter, so the high activity net becomes internal to the cell. Now the high activity node (the output of the AND gate) is driving a much smaller capacitance, reducing dynamic power.

At the bottom of the figure, an AND gate has been initially mapped so that a high activity net is connected to a high power input pin, and a low activity net has been mapped to a low power pin. For multiple input gates there can be a significant difference in the input capacitance - and hence the power - for different pins. By remapping the inputs so the high activity net is connected to the low power input, the optimization tool can reduce dynamic power.

Other examples of gate level power optimization include cell sizing and buffer insertion. In cell sizing, the tool can selectively increase and decrease cell drive strength throughout the critical path to achieve timing and then reduce dynamic power to a minimum.

In buffer insertion, the tool can insert buffers rather than increasing the drive strength of the gate itself. If done in the right situations, this can result in lower power.

Like clock gating, gate level power optimization is performed by the implementation tools, and is transparent to the RTL designer.

## 2.3 Multi $V_{DD}$

Since dynamic power is proportional to  $V_{DD}^2$ , lowering  $V_{DD}$  on selected blocks helps reduce power significantly. Unfortunately, lowering the voltage also increases the delay of the gates in the design.

Consider the example in Figure 2-3. Here the cache RAMS are run at the highest voltage because they are on the critical timing path. The CPU is run at a high voltage because its performance determines system performance. But it can be run at a slightly lower voltage than the cache and still have the overall CPU subsystem performance determined by the cache speed. The rest of the chip can run at a lower voltage

still without impacting overall system performance. Often the rest of the chip is running at a much lower frequency than the CPU as well.

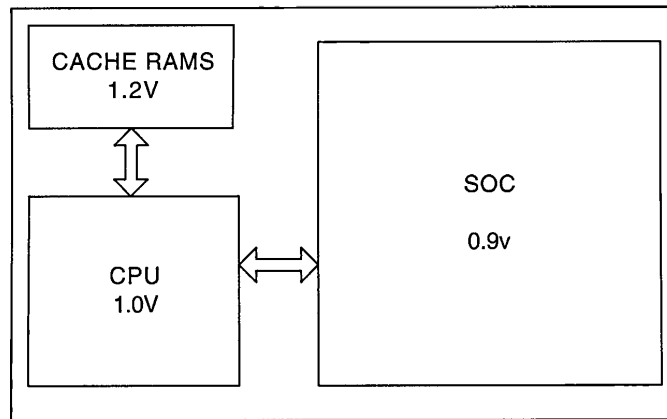


**Figure 2-3** Multi-Voltage Architecture

Thus, each major component of the system is running at the lowest voltage consistent with meeting system timing. This approach can provide significant savings in power.

Mixing blocks at different  $V_{DD}$  supplies adds some complexity to the design – not only do we need to add IO pins to supply the different power rails, but we also need a more complex power grid and level shifters on signals running between blocks. These issues are described in more detail later in the book.

## 2.4 Multi-Threshold Logic

As geometries have shrunk to 130nm, 90nm, and below, using libraries with multiple  $V_T$  has become a common way of reducing leakage voltage.

Figure 2-4 shows the relationship between delay and leakage for a 90nm process. Figure 2-5 shows some representative curves for leakage vs. delay for a multi- $V_T$  library. As explained earlier, sub-threshold leakage depends exponentially on  $V_T$ . Delay has a much weaker dependence on  $V_T$ .

Many libraries today offer two or three versions of their cells: Low  $V_T$ , Standard  $V_T$ , and High  $V_T$ . The implementation tools can take advantage of these libraries to optimize timing and power simultaneously.

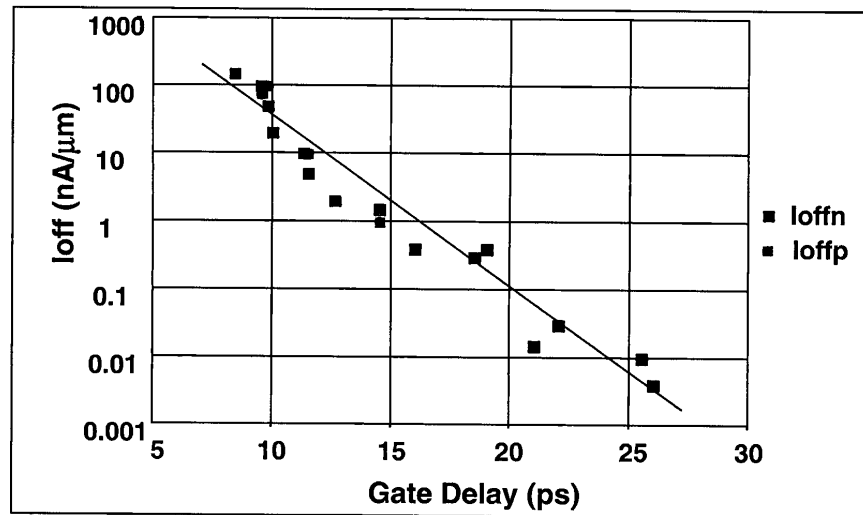


Figure 2-4 Delay vs. Leakage for 90nm

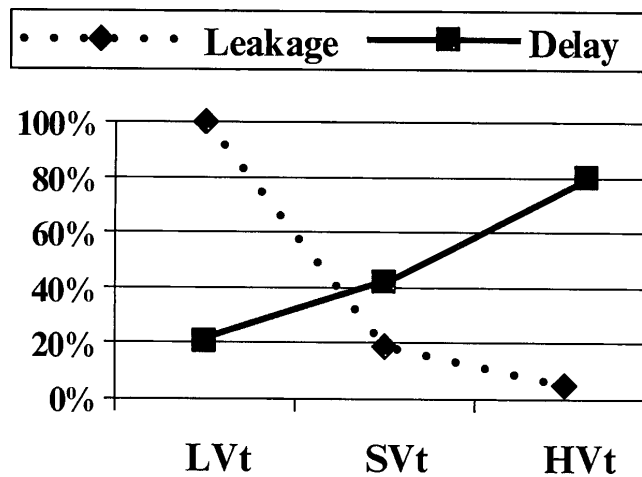


Figure 2-5 Leakage vs. Delay for a 90nm Library

It is now quite common to use a “Dual  $V_T$ ” flow during synthesis. The goal of this approach is to minimize the total number of fast, leaky low  $V_T$  transistors by deploying them only when required to meet timing. This usually involves an initial synthesis targeting a primary library followed by an optimization step targeting one (or more) additional libraries with differing thresholds.

Usually there is a minimum performance which must be met before optimizing power. In practice this usually means synthesizing with the high performance, high

leakage library first and then relaxing back any cells not on the critical path by swapping them for their lower performing, lower leakage equivalents.

If minimizing leakage is more important than achieving a minimum performance then this process can be done the other way around: we can target the low leakage library first and then swap in higher performing, high leakage equivalents in speed critical areas.

## 2.5 Summary of the Impact of Standard Low Power Techniques

Table 2-1 provides a brief summary of the cost/benefit of the techniques described in this chapter.

**Table 2-1**

| Tech-<br>nique   | Power<br>Benefit | Timing<br>Penalty | Area<br>Penalty | Impact:<br>Architec-<br>ture | Impact:<br>Design | Impact:<br>Verifica-<br>tion | Impact:<br>Place &<br>Route |
|------------------|------------------|-------------------|-----------------|------------------------------|-------------------|------------------------------|-----------------------------|
| Multi Vt         | Medium           | Little            | Little          | Low                          | Low               | None                         | Low                         |
| Clock<br>Gating  | Medium           | Little            | Little          | Low                          | Low               | None                         | Low                         |
| Multi<br>Voltage | Large            | Some              | Little          | High                         | Medium            | Low                          | Medium                      |

### References

1. Pokhrel, K. "Physical and Silicon Measures of Low Power Clock Gating Success: An Apple to Apple Case Study", SNUG, 2007 <http://www.snug-universal.org/cgi-bin/search/search.cgi?San+Jose,+2007>.

The SALT technology demonstrator project provided a platform for testing the approaches to power gating and state retention described in this book. In this chapter we give some more details on the system design and RTL coding for this project.

The SALT chip is implemented in 90nm generic technology and contains an ARM processor, an AMBA bus and set of peripherals, and a Synopsys USB OTG digital core and PHY. The ARM core and the USB core are independently power gated. The ARM core uses full state retention; the USB uses partial state retention. Both the ARM core and the USB use switching fabrics of header switches; thus they switch VDD and use a common ground (VSS).

In this first section, we describe the power gating design for the processor. Figure 7-1 on page 86 shows a simplified block diagram of the SALT chip.

## 7.1 Leakage Modes Supported

Most battery-powered processor based designs have to deal carefully with the balance between performance (to support product features) and low power (to support long battery life). The performance requirement steered us to a higher performance, leakier process. To maintain long battery life, we needed to provide aggressive leakage management.

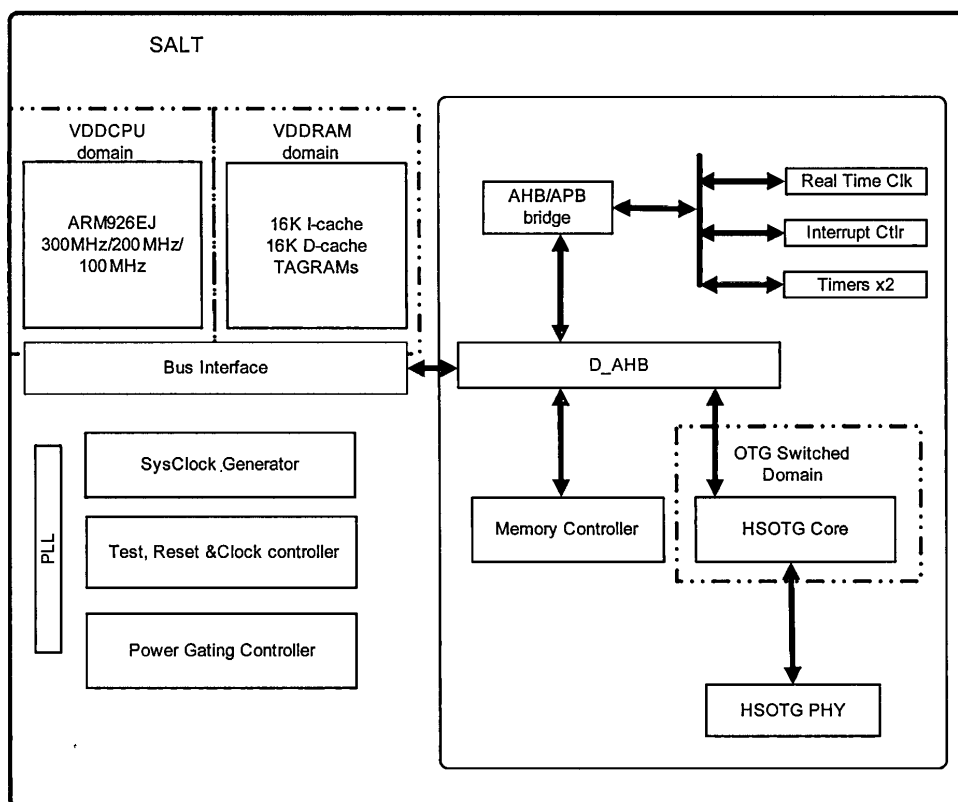


Figure 7-1 SALT Block Diagram

For the SALT project, the processor uses four low-power modes. In all modes, the power controller generates a **SLEEP** signal to enter the low power mode and the **WAKE** signal to exit. In order of increasing leakage savings – and increasing time to power up and power down – the modes are:

- **HALT:** SLEEP turns off the clocks to the processor, WAKE restarts the clocks. Power remains on.
- **SNOOZE:** SLEEP initiates power gating with state retention; power is switched off to the processor itself, but the cache memory remains powered up. WAKE initiates the power up and restore sequence; power is brought up as fast as safety allows.
- **HIBERNATE:** SLEEP initiates a sequence of scanning the processor's internal state to external memory; the VDDCPU power rail is then switched off. WAKE causes the power rail to be switched back on, and the scan chains are used to restore the internal state of the processor. A 32-bit AMBA-based bus is used to write to and read from an external memory. A 32-bit CRC provides integrity checking. The CRC is saved along with the scanned data and used to protect

against restarting with corrupted state. Note that VDDCPU provides power to the processor logic, not cache. So in HIBERNATE mode, only the processor is powered down; the cache memory remains powered up.

- **SHUTDOWN:** SLEEP initiates the same power down sequence as in Hibernate, but now both the processor and the cache have their external power supplies turned off. Explicit code must be called to write back any dirty data in the cache memories before both the VDDCPU and VDDRAM supplies can be power-rail switched. This mode is the only mode not transparent to the operating system.

In addition to these modes, the SALT chip provides active leakage reduction through externally managed threshold scaling. This threshold scaling is done using back-bias control. Both P- and N-wells for the CPU standard cell area are brought out to the chip's pins. This enables experimental analysis of delay and leakage power characteristics as the bias is varied.

Using this well-biasing scheme, we defined three operating modes:

- **NORMAL:** with standard well-bias
- **TURBO-RUN** – with the wells forward biased for increased speed
- **POWERSAVER-RUN** – with the wells reversed biased for reduced leakage

The power controller is designed to ensure that well-bias voltages are only changed while the design is static and un-clocked. Any change between “normal” and back- or forward biased modes of active power management go through one of the HALT, SNOOZE, or HIBERNATE states. Figure 7-2 on page 88 shows the state diagram for the power modes.

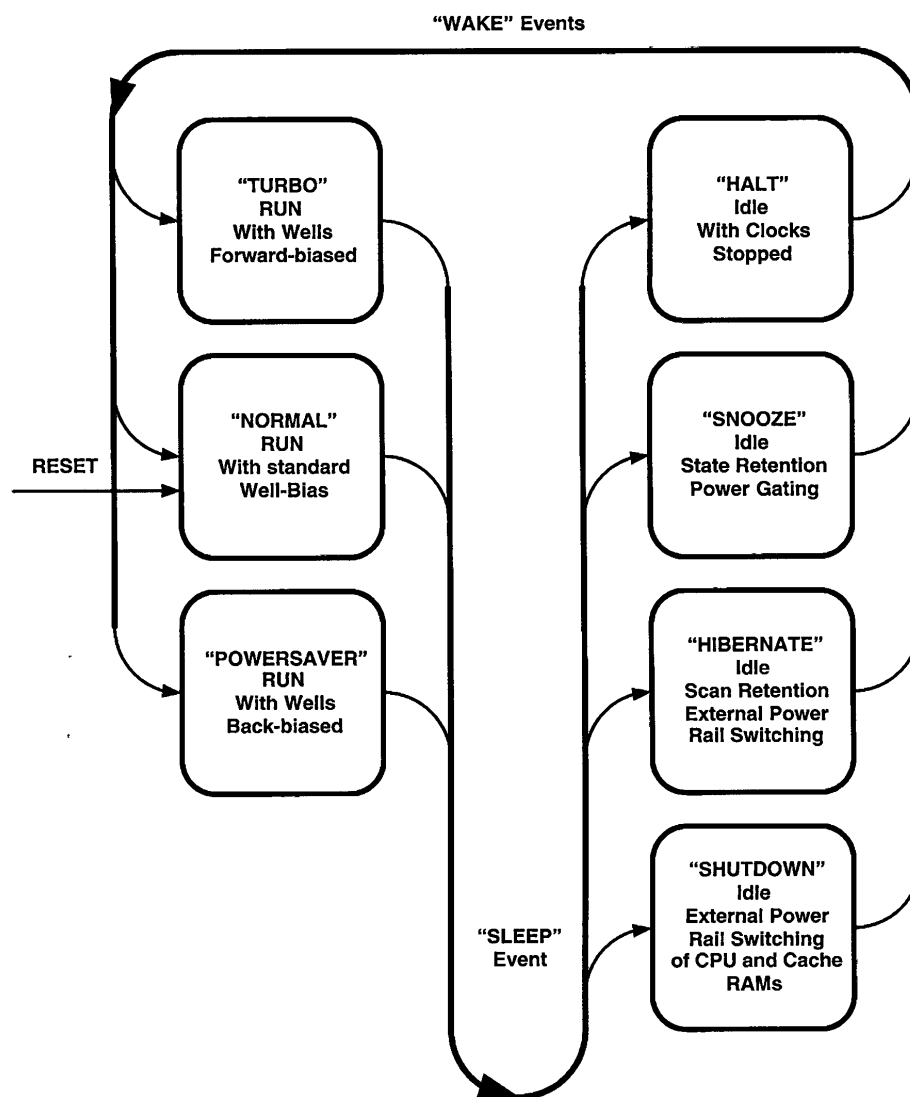


Figure 7-2 Power States for SALT

## 7.2 Design Partitioning

The RTL design is partitioned to allow the three primary power supplies to be mapped to the RTL design:

- **VDDSOC** is the “always-on” supply that powers the entire chip except the processor and its cache.

- **VDDRAM** is an external switched power rail that supplies the Cache and MMU RAMs.
- **VDDCPU** is an external switched power rail that supplies the CPU standard cell area.

VDDSOC provides power to the digital side of the PLLs, the clock generators and the power management control blocks, plus all the real-time peripherals. These peripherals include a real-time clock and timers; these can generate wake-up events as part of their interrupt service requests.

VDDSOC also provides power to the USB OTG subsystem. The USB is power gated though its own power switching fabric using a separate power controller.

Using a separate power supply (VDDRAM) for the cache accomplishes several objectives: it allows the CPU to be powered down while keeping the cache powered, allowing faster restore. It allows the CPU and cache to be powered down while keeping the peripherals of the chip powered up. These peripherals control the wake up of the CPU – upon detecting an interrupt, the interrupt controller signals the power controller to power up the CPU. Finally, using a separate cache power supply allows development of detailed leakage and active power consumption profiles, so that we can determine the minimum operating voltage for the cache memory and the minimum voltage that assures that the cache retains its contents during CPU shut-down.

Similarly, using a separate supply (VDDCPU) for the CPU logic allows detailed leakage and active power consumption profiles to be measured as well as the time and energy cost to get in and out of each power saving state. Figure 7-3 shows the partitioning of the SALT chip.

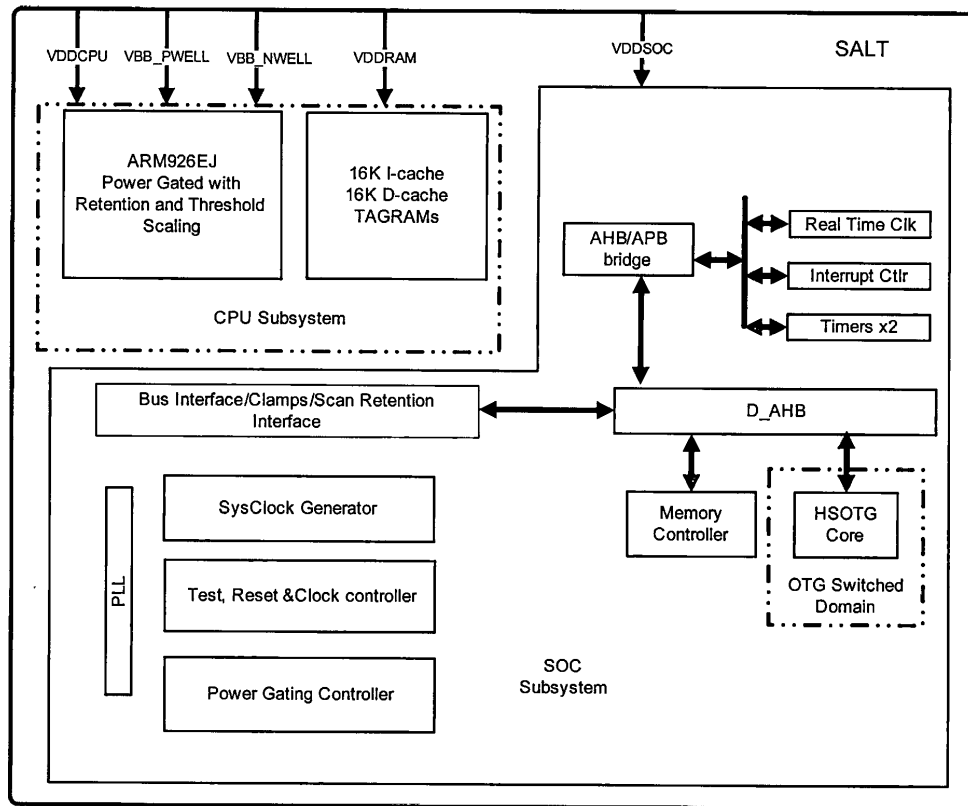


**Figure 7-3** Partitioning of the SALT Chip

Since the original SALT design, the CPU sub-system has been re-implemented to improve design re-use. The Bus Interface Unit and the Power-Gating Controller are now integrated into a “VDDSOC” region within the CPU. This integration significantly reduces the number of signals – and blocks – that the SoC design team has to deal with when integrating the ARM core into the chip.

Although this makes the 3-supply-rail CPU subsystem slightly more complex to implement, the timing and internal power gating and isolation interfaces are now all abstracted away from the top-level SOC design. Any changes or enhancements to the low-power states supported by the IP block are independent of the top level system design.

Figure 7-4 shows the re-partitioned CPU subsystem.

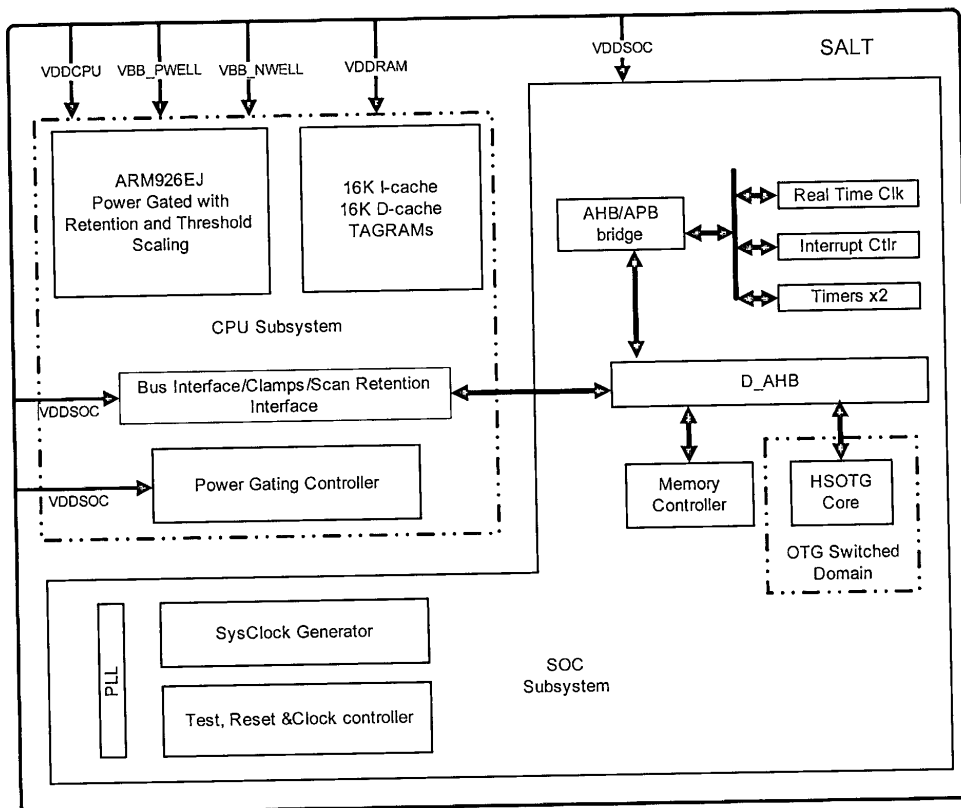


Figure 7-4 Re-Partitioned CPU Subsystem

### Power Gating control and handshakes

The CPU power-gating control system manages:

- the interfaces to the external power supplies for hibernate and shutdown modes
- the bus-clock synchronous scan clock pulses for hibernation-mode save and restore.
- local header-switch power gating control for snooze mode
- isolation and state retention for snooze mode
- the handshake with the system clock generator to switch CPU clock frequencies for the different run modes

One of the goals of the SALT project is to develop a methodology that supports a wide range of library components and current in-rush management techniques. To do this, the CPU power controller uses a request-acknowledge handshake for every control signal: power gating, isolation, save, restore and reset.

All the acknowledge signals are treated as asynchronous to the controller clock, and have local synchronizers to the state machine clock domain.

This approach ensures that the design is free of locally coded delays or counts. It also allows the acknowledge signals to be tied directly to the requests for some implementations or built as true handshakes. In the current SALT implementation, the acknowledges for isolation, save and restore, and reset, are all tied to their respective requests; as implemented, these functions are all very fast and require no delay in sequencing.

The power gating control of the header switches uses a true acknowledge. The power down request is daisy chained through the header switches, so that the acknowledge that arrives back at the controller has a delay equivalent to the switching time of the whole fabric.

A couple of additional design notes:

- After asserting reset, the initialization (power on reset) sequence pulses the save/restore signaling to flush out any X-s from the shadow retention flops. This may be useful when running functional test programs or vectors on the gate level netlist.
- All timing-dependent state machine transitions include a holding term that waits for the output asserted in that state to be acknowledged in order to maintain the timing-independent request/acknowledge sequencing
- The power-gating assertion and de-assertion use a request-acknowledge handshake.
- In the SALT project there are extra diagnostic control inputs to control the switch fabric which allow the power gating to be soft-sequenced or forced fully on and off, and only the power-gating acknowledge input to the state machine is used to determine when power is safely restored.

## 7.3 Isolation

The SALT project uses several different isolation techniques. The initial version of SALT was done before there was tool support for automatically inserting isolation cells. As a result, we manually inserted these cells in the RTL.

The interface between the CPU and cache is particularly timing critical, requiring careful design and timing analysis. We put isolation cells on all of the outputs from the CPU to the cache, but placed them in the cache's VDDRAM region. This was convenient in the SALT design because the cache is always powered up, making power routing to the isolation cells simpler.

The VDDRAM region has input isolation cells instantiated as “Generic Library Cells.” These are just RTL wrappers that can be mapped either to behavioral simulation models or to technology-specific clamp cells from the “Power Management Kit” for the standard cell library. This approach provides explicit instantiation of the cells on the many critical path signals, including clocks and resets, into the memories from the CPU core logic. Instantiating these cells in the RTL provides clean visibility of the clocks and critical signals that need to be managed carefully in the implementation flow. Clock balancing across isolation cells is not straightforward; isolation cells typically limit the flexibility the clock buffering tools in restructuring the buffer trees.

The outputs from the CPU to the Bus Interface Unit are isolated by cells placed in the CPU – that is, in the VDDCPU region. These output isolation cells pull-down all output signals at the AMBA bus interface in Snooze mode (local power gating), to guarantee clean SOC interface signals.

However when the CPU rail is switched off (Hibernate mode) these isolation cells lose their VDDCPU power and the outputs could again float. To address this problem, simple bus repeater or hold cells were added in the VDDSOC domain to clamp these outputs while avoiding any further gate delays. In addition, the isolate control signal from the power controller drives reset in the bus interface module to force logic-0 clamping of all bus interface protocol signals.

The power controller generates a single isolate control signal, but it is routed as two separate signals. One copy goes to the cache RAM without going through the CPU region, and one copy goes to the CPU without going through the cache. When VDDCPU has its power rail shut down during hibernate, the isolate signal for the cache is still powered up since all the buffers in its path are powered by VDDSOC or VDDRAM. Thus, the inputs of the cache are protected from floating signals (and data corruption) during hibernate.

The USB OTG block uses the alternative of instantiated AND gate cells in the RTL with suitable “don’t-touch” attributes added to prevent logical optimization across these isolation boundaries. During synthesis we force these AND gates to be mapped to cells that do not crowbar with one floating input (as long as the other input is low).

All of the isolation cells in the SALT design were instantiated using a generic, technology-independent wrapper module. For hand instantiated cells, this provides a degree of design portability and enables simulation before a specific technology library has been selected.

The initial version of the SALT chip instantiates isolation cells in the RTL as described above. In a subsequent version, we use EDA tools to implement the isolation cells without modifying the RTL. The re-partitioning of the CPU subsystem to include VDDSOC also eliminates the need for bus repeater cells for the AMBA bus

interface signals. Now the isolation cells get power from VDDSOC, which is always on. Thus, a single set of isolation cells is used to isolate the bus interface signals.

With the UPF-defined simulation semantics, it is possible to instantiate and simulate isolation cells by issuing a UPF tcl command (`set_isolation`); no hand instantiation or generic wrappers are required.

## 7.4 Retention

The SALT project incorporated several retention techniques to allow comparison and analysis of the area/time/energy cost functions for each technique on the same silicon.

The CPU uses full state retention. Given a fully validated CPU core, retaining every register bit state is the only safe approach to ensuring that the processor can be restarted with arbitrary control and data state. To retain only the architectural state, and use reset signal for non-architectural state, would require a serious verification project.

A key concern of any power gating design is to assure that retention registers do not get corrupted during the power down/power up sequence. On SALT we added a non-real-time diagnostic mechanism to the power control sequencing. It uses the “Hibernate” scan functionality to check-sum and save to external memory the contents of the all the shadow registers after the SAVE operation and then checksum and save the entire register contents after the RESTORE operation. These can then be compared to detect any corruption of the retention registers during power sequencing.

This approach allows error analysis for both random and location sensitive problems and the efficacy of the soft-start power gating sequencing in limiting in-rush current. This also turned out to be a valuable way of quantifying the safety margins of the retention flops and allowed them to be subjected to thermal and voltage shocks while in retention mode.

On the other hand the USB OTG core uses partial retention. The core is partitioned with persistent USB end-point data held in the Control and Status Register (CSR) block, while the Protocol Interface Engine (PIE) block contains only protocol state. Data for the current transaction is held in a RAM-based FIFO. During power down, the USB waits for the current transaction to finish, so the FIFO is empty. It then saves all the state of the CSR, using the standard retention register technique. On power up, the CSR has its state restored, and a reset is issued to the PIE. This resets the protocol engine, and it is ready for the next transaction.

The USB power gating design is discussed in more detail in Chapter 8.

## 7.5 Inferring Power Gating and Retention

In the SALT project, we use a retention register cell with a single-pin control to control save and restore in an edge-triggered manner. Retention state is captured on the falling edge of an active-low **NRETAIN** signal and restored on rising-edge of the same **NRETAIN**:

Building on the same worked example described in Chapter 5 we can modify the UPF to show this retention behavior as follows:

```
set_retention_control my_retention_strategy
    -domain my_power_domain
    -save_signal {NRETAIN negedge}
    -restore_signal {NRETAIN posedge}
```

Or, if we are using a simulator that does not support UPF, we can modify the RTL to show retention behavior as follows:

```
`ifdef RTL_PG_EMULATE
    reg [3:0] state_SAVE = 4'bXXXX;
    wire PWR;
    assign PWR = pwr_req & pwr_ack;
`endif

always @ (posedge clk or negedge nrst
`ifdef RTL_PG_EMULATE
    or negedge PWR or negedge NRETAIN
`endif
) begin
    `ifdef RTL_PG_EMULATE
        if (!PWR)
            current_state <= 4'bXXXX;
        else if (!NRETAIN)
            current_state <= state_SAVE;
        else
            `endif

        if (!nrst)
            current_state <= 4'b0101;
        else
            current_state <= next_state;
    end
```

```
`ifndef RTL_PG_EMULATE
always @ (negedge NRETAIN)
    state_SAVE <= current_state;
`endif
```

## 7.6 Measurements and Analysis

We evaluated the SALT project silicon to understand the power gating and state retention improvements to leakage power, and the effect of the switching fabric on the functional performance.

Figure 7-5 shows the basic evaluation of the dynamic and leakage power as measured for both the cache memories and the standard cell logic for the ARM926EJ CPU subsystem stepping the power supply in 10% steps from 110% of nominal down to 70% of nominal. For this 90nm Generic process technology the nominal supply voltage is 1.0V.

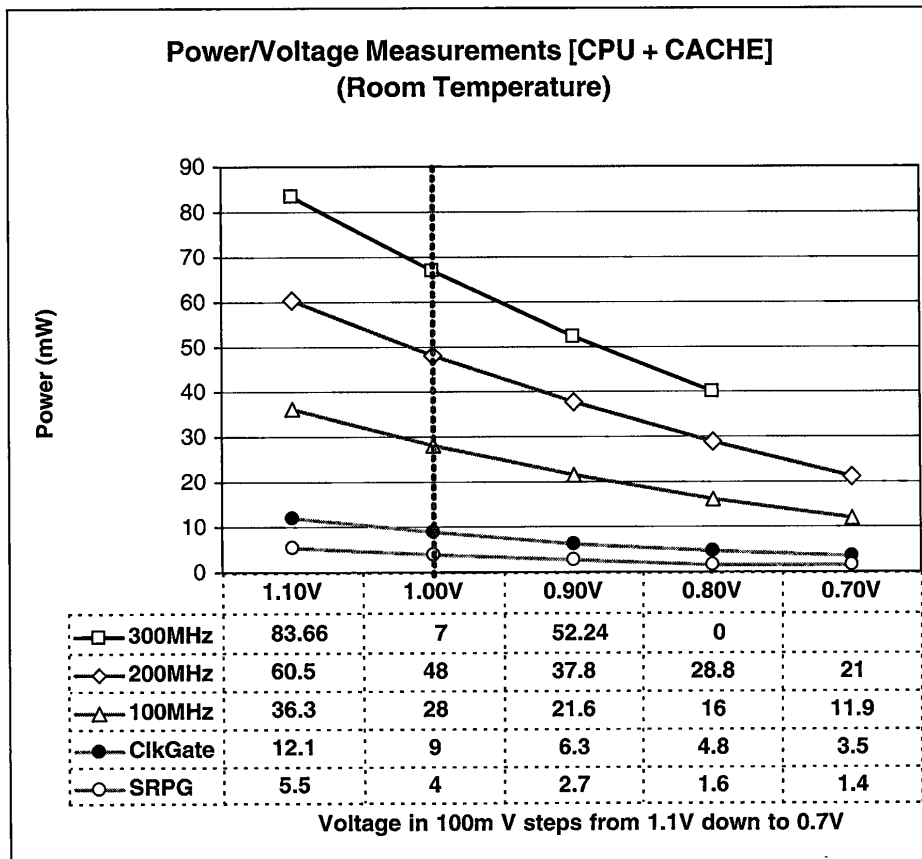


**Figure 7-5** Power Measurements for SALT

The measurements are at the circuit board level – before any package/bond-wire/on-chip IR drop and the power gating fabric itself – so are very reasonable for “typical” silicon at room temperature.

The worst-case timing sign-off target frequency for the CPU design was 300MHz. The first three measurements show the dynamic power with the CPU operating at 300MHz, 200MHz, and 100MHz respectively.

The “ClkGate” measurements reflect the baseline leakage when the clock for the entire CPU is gated off.

The “SRPG” (Save Restore Power Gating) measurements are the leakage during power gating. Thus it reflects the leakage of the switches, the retention latches and the always-on control buffer trees, and in this case also includes the baseline leakage for the cache memories that are not power-gated.

Figure 7-6 shows the measured leakage power for the power-gated “VDDCPU” power domain across the temperature range of key interest for battery-powered products. The vertical scale is plotted logarithmically:

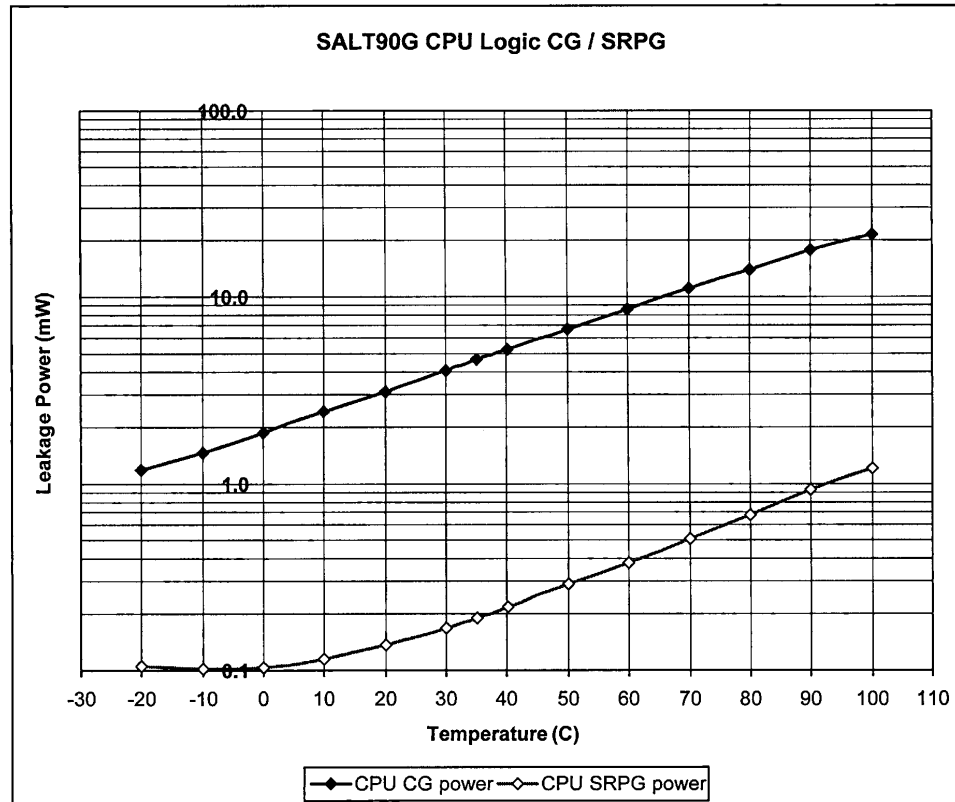


Figure 7-6 Leakage Power on SALT

The upper curve plotted is the baseline leakage when the clocks are stopped. The lower curve is the measured leakage power with State-Retention Power-Gating. Across the entire temperature range the leakage power savings are more than 10-fold and greater than 25-fold around room temperature.

Figure 7-7 shows the measured leakage power for not only the power gated standard cell logic but also the cache memories, which were not available with power-gating for this project:

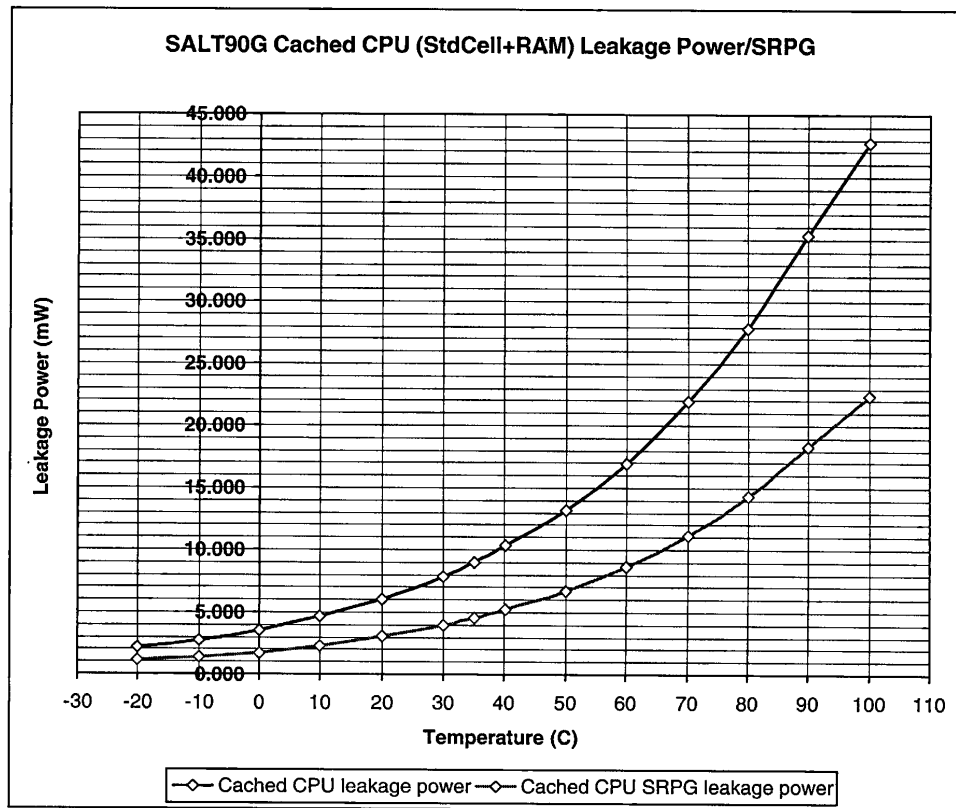


**Figure 7-7** Leakage vs. Temperature

The upper curve is the baseline total leakage for the CPU subsystem (logic plus RAMs) and the lower curve shows the power with the logic portion power gated with state retention.

The leakage power of the RAM now dominates the leakage power but there is still a factor of two leakage power saving for the CPU subsystem. With integrated power gated RAMs the savings could be improved to closer to those achieved for the logic portion.

Figure 7-8 on page 100 shows the detailed comparison of the leakage power savings of state retention relative to the baseline leakage. The ratio of baseline leakage to SRPG leakage is plotted, on a linear scale, to show how this characteristic varies across the temperature range.

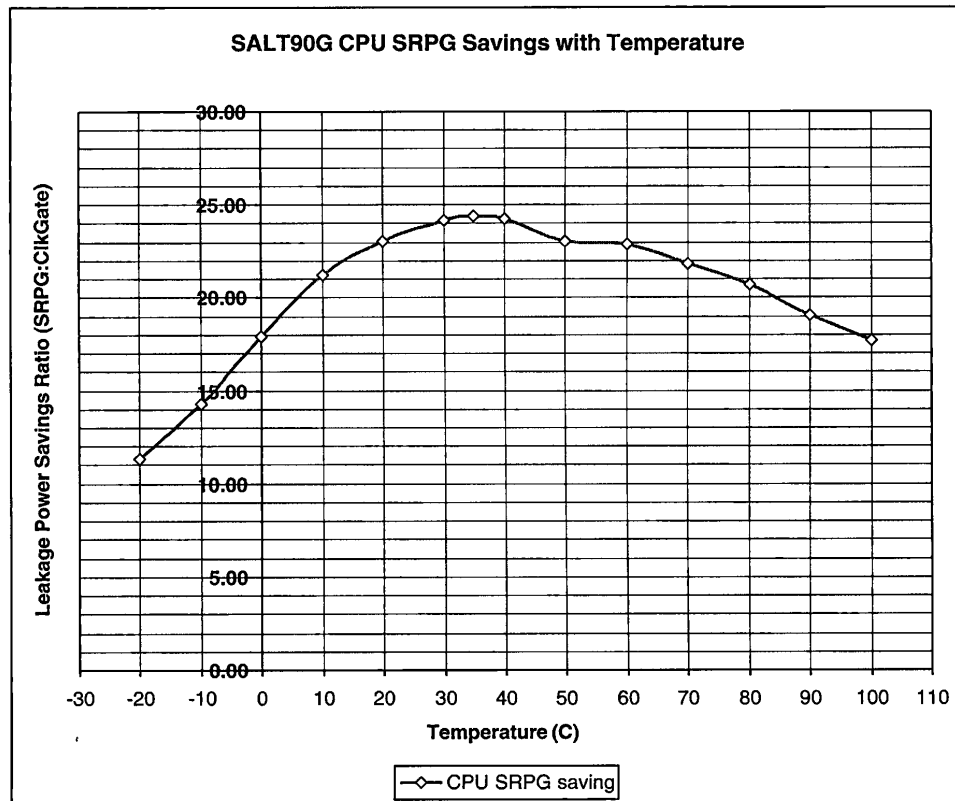


**Figure 7-8** Leakage Savings vs. Temperature

The deterioration of the savings at the high end of the temperature range can be explained by the leakage behavior of the High- $V_T$  switches. The very best power-gating leakage power saving for this particular design and technology was achieved at 35C.

In summary, power gating techniques when applied to standard-cell logic produced leakage power savings of between 10x and 25x over the baseline leakage. Because cache memories are typically tuned for performance and so exhibit a fairly high leakage power, these too benefit from leakage reduction techniques. This topic is addressed in Chapter 12.

The SALT technology demonstrator has been an effective vehicle for developing State-Retention Power-Gating techniques and methodologies and for analyzing the costs and benefits of power gating on a 90nm process. These techniques and methodologies will be highly appropriate at 65nm and below.

## *Implementing Multi-Voltage, Power Gated Designs*

This chapter describes the implementation of designs that use power gating and multi-voltage techniques. It highlights the areas in the implementation process that are specific to multi-voltage and power gating designs.

To illustrate the implementation process, we use a multi-voltage power gated ARM1176JZF-S microprocessor as an example design.

The ARM1176JZF-S integrates a number of technologies including power gating and dynamic voltage scaling. It employs the IEM techniques described in Chapter 9; it dynamically monitors and predicts the performance requirements of multiple applications, and tunes the processor's operating voltage and frequency to match the requirements. These techniques reduce the processor's energy consumption by 25% - 50%. In order to exploit this IEM technology, the ARM1176JZF-S processor has been architected for a low power implementation.

The ARM1176JZF-S is illustrated in Figure 11-1 on page 156 and consists of a cache sub-system, central core CPU, memory management sub-system and AXI interface.

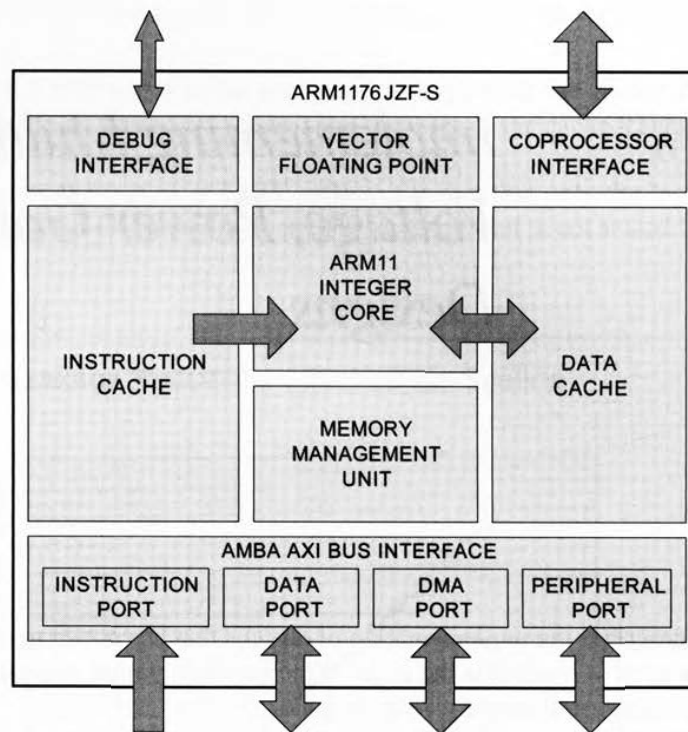


**Figure 11-1** ARM1176JZF-S Synthesizable Applications Processor

There are three power domains within the ARM1176JZF-S

- An always-on power domain (VSOC) containing the logic that interfaces to the SoC through the AXI interface. The logic in this power domain also manages the asynchronous mode of the processor when performance scaling is employed.
- A shutdown power domain (VCPU) that contains all of the core CPU logic. This power domain can operate at multiple voltages and can also be powered down.
- An always-on power domain (VRAM) that contains the cache memory instances. Typically, memory cannot be scaled to the same degree as standard cells and during voltage scaling the VRAM and VCPU power domains need to be scaled independently. Also, when VCPU is shutdown, the VRAM power domain can be placed into retention. However, as discussed earlier in this book, if the cache is placed into retention state during sleep then the wake-up from sleep will take longer since the time for the power supply to the memories to stabilize will need to be factored into the wake-up time.

Figure 11-2 below shows the basic structure of the ARM1176JZF-S from a power domain perspective.

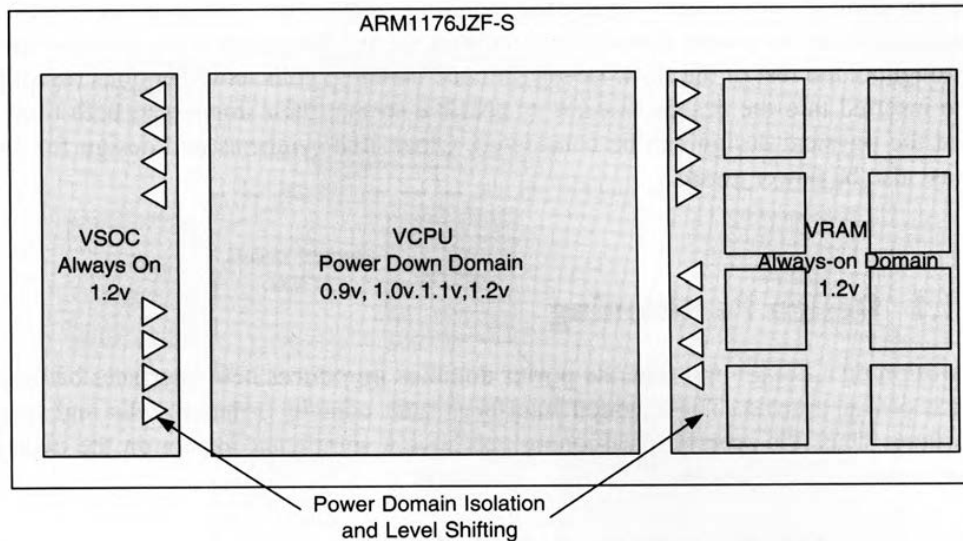


**Figure 11-2** ARM1176JZF-S Multi-voltage, power gated design

VSOC always operates at full voltage – the same voltage as the logic sitting on the AXI bus externally to the processor. The VCPU power domain can operate at a number of voltages and can be shutdown. VRAM operates at the same voltage as VSOC or is placed into retention if VCPU is shutdown. Placing a memory into its retention state typically means stopping the clock(s), disabling the memory and reducing its supply voltage to the retention voltage.

To implement this design, we use an implementation process that allows for the logical and physical partitioning of multiple power domains with multiple voltage levels. The implementation process also supports the insertion of level shifters, isolation cells, and retention flops. We will also provide power switching networks for the VCPU power domain. As we proceed with the implementation, we pay particular attention to management of voltage drop, impacts on timing from our multi-voltage partitioning and the electrical integrity of our design, particularly at the power domain interfaces.

For the discussion in the rest of this chapter, we introduce two definitions:

A *Power Domain* is a logical entity and is a collection of design elements that share a primary power supply.

A *Voltage Area* is a physical entity that represents a geographic area of the chip that is used to place the logic within a given power domain. It is possible for a power domain to have a number of disjoint voltage areas depending on the nature of the chip. However, in the discussion that follows, it is assumed that there is a one to one relationship between a voltage area and a power domain.

Power domains are created during the synthesis phase; the voltage areas (physical realizations of the power domains) are created during design planning and managed throughout the rest of the flow. Levels shifters, isolation cells and retention registers are inserted into the design as early as possible so that their impacts to both timing and the physical design can be considered. Clock tree synthesis and design for test must also be power aware.

## 11.1 Design Partitioning

Partitioning a design into separate power domains introduces new interfaces between each power domain. These added interfaces may contain isolation cells and level shifters. Thus, the specific partitioning can have a significant impact on the overall performance of the design.

The job of partitioning a design into power domains is a joint responsibility of the system architect(s), RTL designer(s), and implementation engineer(s). Between them they understand the target application, the power and performance objectives, and the limitations of the target technology.

### 11.1.1 Logical and Physical Hierarchy

When mapping the logical hierarchy to power domains, we want to assign hierarchical functional units in their entirety wherever possible. Synthesis engines operate top down on a cell in the hierarchy and the best Quality of Results (QoR) is achieved if the entire hierarchy can be manipulated under the same goals and constraints.

Gathering up various hierarchical cells from different levels of the design and assigning them to power domains can make timing closure significantly more difficult.

Figure 11-3 shows the logical hierarchy of the ARM1176JZF-S after some logical grouping of top level modules in preparation for a low power implementation.

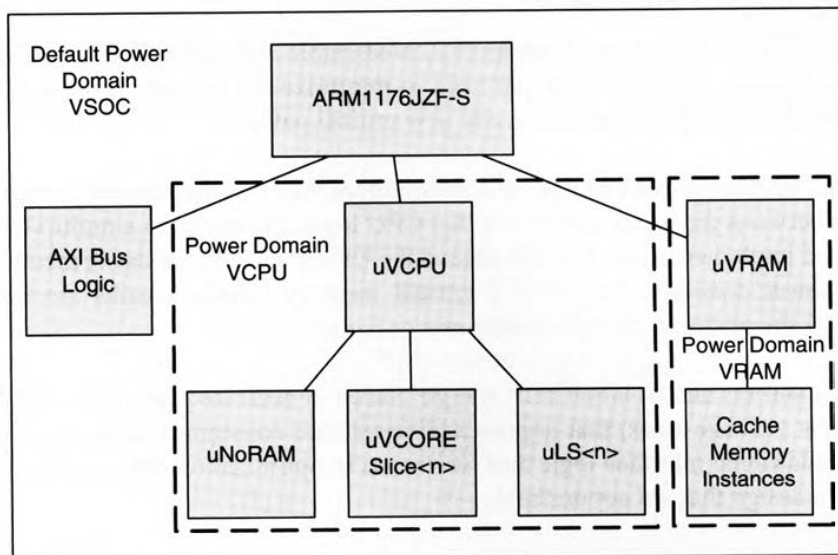


**Figure 11-3** Alignment of power domain to logical hierarchy

In this example, we map the entire CPU to the VCPU domain, the whole cache – logic and RAMS – to the VRAM power domain, and the AXI bus to the VSOC power domain.

This partitioning meets several objectives:

- In some power modes, we power gate the CPU, but keep the cache powered up to enable a fast restart.
- For multi-voltage, we will operate the CPU at several different voltages, depending on work load. But in many technologies there is not enough headroom to run the cache RAMs at these lower voltages.

By maintaining a high correlation between the power domain structure and the logical hierarchy we can minimize the number of signals that cross the power domain interfaces. This minimizes the amount of logic we need to introduce to maintain electrical integrity. This in turn minimizes the QoR impact to our design and makes physical implementation easier.

For the same reasons, we have implemented the minimum number of power domains required to meet our low power objectives. Creation of unnecessary power domains makes the implementation process more complex and can negatively impact QoR.

### 11.1.2 Critical Path Timing

In most designs, the critical paths are known, predictable and well understood. During design partitioning for low power, we need to minimize the impact on these critical paths and make sure we do not create any new critical paths.

In our case, we have a cached microprocessor, where the critical paths are through the interfaces between the cache system and the CPU logic. However, in a multi-voltage, power gated implementation, we have placed the CPU logic and the cache memory in separate power domains. Our already critical memory interface paths are further impacted by the addition of level shifters and/or isolation logic.

These level shifters and isolation cells will be placed in dedicated geographic areas in the floorplan (voltage areas) that impose additional hard constraints on any optimization. This additional interface logic and restriction in optimization may create critical paths in the design that are not desirable.

This is also true for off-chip interfaces. Any logic in the design that is operating at a voltage level different to that of the IO will need to be level shifted prior to the IO interface thereby adding additional delay to these off-chip paths.

When partitioning a design into multiple power domains we also need to avoid the situation where logic in always-on blocks is dependant on state set in powered down blocks.

Investing the time to architect power domains carefully can help minimize the impact on physical design and system timing.

Recommendations:

- Ensure that the power domain hierarchy aligns with logical hierarchy where possible
- Pay particular attention to the critical path timing in the design – especially around power domain interfaces

## 11.2 Design Flow Overview

As show in Figure 11-4, the design flow for a multi-voltage, power gating design follows that of a standard implementation with a few exceptions.

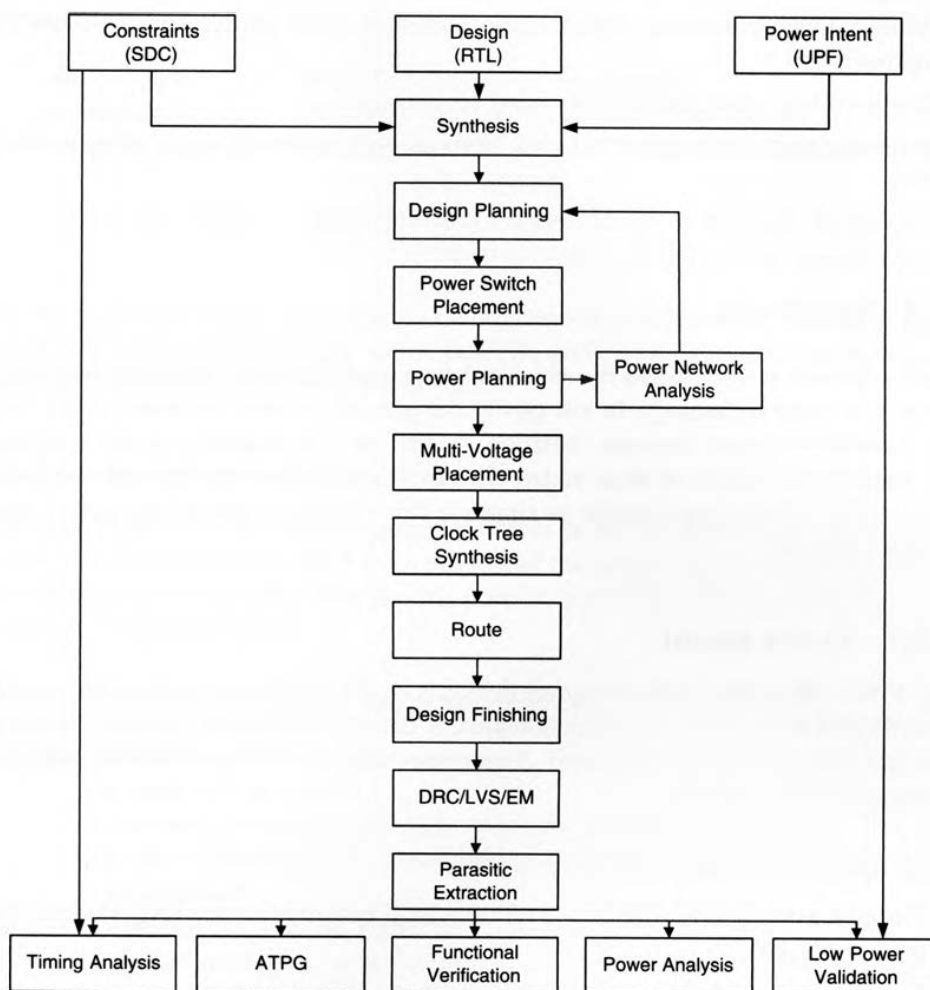


**Figure 11-4** Design Flow Overview

Primarily, the differences between the standard design flow and that for a multi-voltage, power gated design are:

- Specification of the power intent
- Creation of power domains during synthesis
- State retention synthesis with always-on network management
- Multi-voltage physical design partitioning and the addition of MTCMOS switch cells
- Multi-voltage power network synthesis for MTCMOS power gating during power planning phase
- Early power network analysis to validate power gating switch topology

- Multi-voltage placement optimization including level shifter and isolation cell optimization
- Multi-voltage clock tree synthesis and optimization
- Intensive post route power network analysis with power-up sequencing verification

## 11.3 Synthesis

Logic synthesis is the process by which we map a generic RTL representation of the design to a target technology. In low power designs, we include the power intent – the specification of power domains, isolation, and so on – at synthesis so that optimization includes the effects of these added circuits. We can view the logic synthesis of a multi-voltage power gated design as simply a low power overlay to the typical logic synthesis process.

### 11.3.1 Power Intent

This power intent describes the power domains in the design and defines how power is distributed to the various power domains. It defines where level shifters, isolation cells and retention flops are required. It describes how power is switched on and off to the various power domains.

Power intent can be captured in several ways

- Through explicit definition in the RTL by hand instantiating the level shifters, isolation cells, etc.
- Through explicit definition in the RTL by using HDL pragmas.
- Through the use of tool-specific commands.
- Through the use of an accompanying power file in a standard format such as the Unified Power Format (UPF).

For the purposes of the discussion here, we will use the UPF format.

### 11.3.2 Defining Power Domains and Power Connectivity

In UPF, power domains are specified with the following command.

```
create_power_domain domain_name  
    [-elements list]  
    [-include_scope]  
    [-scope instance_name]
```

Where:

|                                |                                                                                                         |
|--------------------------------|---------------------------------------------------------------------------------------------------------|
| <i>domain_name</i>             | Identifies the new power domain                                                                         |
| <b>-elements</b> <i>list</i>   | Indicates which elements are in the domain.                                                             |
| <b>-include_scope</b>          | Indicates that the scope of the domain should be included in the domain.                                |
| <b>-scope</b> <i>inst_name</i> | specifies that the scope (instance) in which the power domain is created. Default is the current scope. |

As discussed previously, it is strongly recommended that the power domain hierarchy align with the logical hierarchy where possible and that the elements list be a collection of hierarchical cells within the design.

### 11.3.3 Isolation Cell Insertion

Isolation logic must be inserted at the interfaces of power gated blocks such that the logic in these blocks is isolated from the rest of the design during shutdown. Specific isolation cells are typically used for this purpose and these cells are available in most physical IP libraries today.

In UPF, isolation cells are specified by the following two commands:

```
set_isolation isolation_name
    -domain domain_name
    <-isolation_power_net net_name | -isolation_ground_net net_name |
    -isolation_power_net net_name -isolation_ground_net net_name |
    -no_isolation>
    [-elements list] [-clamp_value <0 | 1 | latch | Z>]
    [-applies_to <inputs | outputs | both>]
```

Where:

|                       |                                                                                                                 |
|-----------------------|-----------------------------------------------------------------------------------------------------------------|
| <i>isolation_name</i> | specifies the isolation strategy name.                                                                          |
| <i>domain_name</i>    | is the name of the domain to which the strategy is to be applied.                                               |
| <b>-elements</b>      | indicates which elements of the domain will have their interfaces isolated.                                     |
| <i>net_name</i>       | identifies the supply net(s) used to supply the isolation logic inferred by this strategy.                      |
| <b>-no_isolation</b>  | does not isolate the port, pin, or design element specified in the elements list.                               |
| <i>clamp_value</i>    | is the value to which the input or output shall be clamped. The default is 0.                                   |
| <i>applies_to</i>     | indicates whether the domain's input ports, output ports, or both are isolated. The default is <b>outputs</b> . |

We recommend isolating all outputs from a shutdown block. These recommendations can be met by using the default value for the **applies\_to** and **-elements**.

Having defined an isolation strategy for a shutdown block, we must now also define how the isolation is controlled. The **set\_isolation\_control** command does this.

```
set_isolation_control isolation_name
    -domain domain_name
    -isolation_signal signal_name
    [-isolation_sense <high | low>]
    [-location <self | parent | sibling | fanout | automatic>]
```

Where:

|             |                                                                                                          |
|-------------|----------------------------------------------------------------------------------------------------------|
| signal_name | identifies the signal that causes the specified element to drive its clamp value.                        |
| -location   | specifies where the isolation cells are placed in the logic hierarchy. The default is <b>automatic</b> . |

Location determines where the level shifters will be inserted:

- Self means in the module whose output is being shifted
- Parent means in the parent of the module whose output is being shifted
- Sibling means a new module is created at the same level as the module whose output is being shifted. The shifters are put in this new module.
- Fanout means that the shifters will be placed in all the destination modules of the shifted signals.
- Automatic means that the tools are free to put the shifters in the appropriate location.

There can be multiple styles of isolation cell each with specific requirements on where they are placed and how they are power routed. Thus, we recommend that the location of these cells be determined automatically by the synthesis tool based on the nature of the isolation cell being targeted. The default value of **automatic** for the **-location** switch accomplishes this.

### 11.3.4 Retention Register Insertion

Retention registers are synthesized automatically during the logic synthesis phase once the retention style and associated attributes have been defined.

In UPF, the state retention strategy and the specification of state retention control are separated and are specified by the following commands.

```
set_retention retention_name
    -domain domain_name
    <-retention_power_net net_name | -retention_ground_net net_name |
    -retention_power_net net_name -retention_ground_net net_name>
    [-elements list]
```

Where:

|                             |                                                                                                                     |
|-----------------------------|---------------------------------------------------------------------------------------------------------------------|
| <i>retention_name</i>       | specifies the retention strategy name                                                                               |
| <i>domain_name</i>          | specifies the domain for which this strategy is applied.                                                            |
| <i>net_name</i>             | identifies the supply net(s) used to supply “always-on” power to the retention registers inferred by this strategy. |
| <b>elements</b> <i>list</i> | indicates which objects in the power domain.                                                                        |

Selecting a subset of elements in a design which shall have their state retained during shutdown is an architectural decision and should not be made during the implementation phase. Implementing partial state retention in a design must be a carefully managed process to ensure that the design can come out of a sleep mode correctly. The interaction and protocols between state retention functionality and the system reset controls must be well defined and honored during wakeup.

Therefore, the list of elements provided to the **set\_retention** command is an integral part of the design architecture specification and should be identified prior to the implementation phase.

Having defined the retention strategy it is now necessary to define the retention control network.

```
set_retention_control retention_name
    -domain domain_name
    -save_signal {{net_name <high | low | posedge | negedge>}}
    -restore_signal {{net_name <high | low | posedge | negedge>}}
```

Where:

|                    |                                                                                                 |
|--------------------|-------------------------------------------------------------------------------------------------|
| <i>save_net</i>    | identifies the signal that causes the register values to be saved into the shadow registers.    |
| <i>restore_net</i> | identifies the signal that causes the register values to be restored from the shadow registers. |

The specific values for the options in this command are again related to the architecture of the design and the specific retention registers available in the library.

Note: The `set_retention_control` command also has optional arguments for defining assertions about the save and restore signals to assist in verification. See the full UPF specification of details on these arguments.

### 11.3.5 Level Shifter Insertion

Level shifters are inserted automatically during synthesis once the location and usage rules have been defined.

In UPF, level shifters are specified by the following command:

```
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 | sibling | fanout | automatic>]
    [-no_shift]
```

Where:

|                           |                                                                                                                                                                                                    |
|---------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>level_shifter_name</i> | is the name of the level shifter strategy, which is used by the tools for reporting.                                                                                                               |
| <i>domain_name</i>        | is the name of the domain to which the strategy is to be applied.                                                                                                                                  |
| <b>elements</b>           | indicates which elements of the domain will have their interfaces shifted. Default is all the interfaces.                                                                                          |
| <b>applies_to</b>         | indicates whether to shift the (elements of the) domain inputs, outputs, or both. The default is both.                                                                                             |
| <b>threshold</b>          | defines the voltage, in Volts, for determining when to insert level shifters. If the difference between two domains is greater than the threshold, level shifters are inserted. The default is 0V. |
| <b>rule</b>               | determines whether to insert level shifters for interfaces that go from a lower voltage to a higher one, a higher one to a lower one, or both. Default is both.                                    |

We strongly recommended that level shifters be used on all power domain interfaces where an up shift is required, since this prevents crowbar currents and improves edge rates and therefore timing.

In general, we recommend that level shifters be used on all power domain interfaces where a down shift is required. It is safe to overdrive the input in the lower voltage domain. However, the timing characteristics of the destination cell will assume a correctly driven input signal operating at the same voltages the output driver of the cell. There will be an error in calculating the delay if a level shifter is not used.

Thus, we recommend **-rule both**.

The placement of the level shifters is important. The output driver for a level shifter requires more supply current than the input stage. For this reason, we recommend placing the level shifter in the destination domain – the domain that the level shifter output drives. This assures a high quality power connection to the output stage of the level shifter cell.

There are two ways to do this with the **set\_level\_shifter** command. We can say:

**applies\_to inputs**  
**-location self**

Or we can say:

**applies\_to outputs**  
**-location fanout**

There is no general, ironclad rule to prescribe the difference in voltage levels above which level shifters are required. This decision is technology and library dependant.

In certain cases, when the voltage difference between the two power domains is less than the threshold voltage then level shifters are not strictly required. However, when making this decision, the tolerance of the power supplies should be considered. It may be the case that when both power domains are being powered by ideal supplies that the voltage difference is tolerable, however when worst case variation between the power supplies is considered, the difference may be too great and level shifting required.

Using the default **-threshold** of 0V is a safe initial value. If timing across critical interfaces becomes a problem, we can revisit this issue and specify a different value.

In any case, we do need to specify unambiguously which power domains are at what voltage level, so the tools know where to insert level shifters. The UPF power state

table constructs (`create_pst` and `add_pst_state`) provide an explicit mechanism to specify voltage levels.

In the diagram below, a net is entering the VSOC power domain which is at a different voltage from the cell driving this net. In this situation, the net must not be buffered (in the SOC domain) prior to the level shifter. Otherwise we defeat the whole purpose of using a level shifter. The requirement to avoid buffering of level shifter input effectively forces the level shifter cells to be placed close to a power domain boundary. Modern tools address these issues automatically.

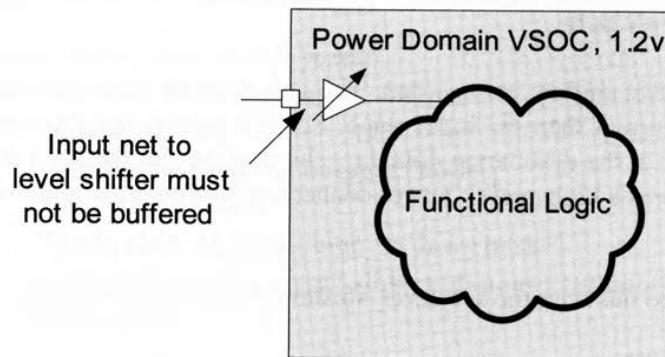


**Figure 11-5** Input Nets on Level Shifters are Protected

#### Recommendations:

- Define a comprehensive power intent for the design that can be implemented and verified
- Define the power domains early and manage the power intent throughout the entire design process
- Define isolation logic and state retention as part of the design description (in the RTL or UPF file) to ensure full design verification
- Include isolation and state retention synthesis in the overall logic synthesis step – not as an afterthought. The area and delay cost of isolation and state retention should be an integral part of the overall cost function for the design.
- Ensure that all control networks for always-on logic (switch cells and retention registers) are buffered using always-on buffers to ensure that they remain alive during power down.

### 11.3.6 Scan Synthesis

As described in Chapter 5, low power designs pose special problems for scan testing.

The typical DFT challenges faced in multi-voltage design include:

- Power aware architecture and reordering of scan chains
- Automatic insertion of level shifters and isolation logic on DFT signaling across power domains
- Timing issues in scan chains that cross power domains operating at multiple voltage levels
- Intelligent routing of the scan enable signal(s) to minimize power domain crossing

Figure 11-6 shows the situation we want to avoid.

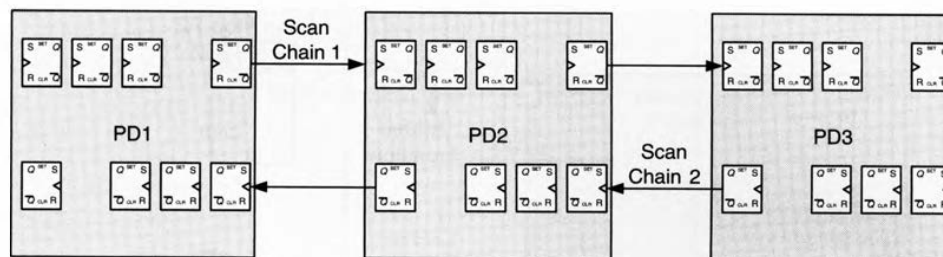


**Figure 11-6** Power Domain Mixing of Scan Chains

This situation, where scan chains cross power domains, can cause real problems. At the least, level shifters and isolation cells may have to be inserted.

In the worst case, keeping all three power domains powered up during test is not possible. Since switching activity in scan mode can be much higher than in normal operation, scan testing with all domains powered up may exceed the max power for the chip.

For these reasons, we need to make every effort to implement the scan chain structure in a power aware style, where the scan paths are localized within each power domain. The scan elements can then be reordered within the power domain during placement optimization to minimize wire length. We also avoid requiring the clock skew on the scan paths to be balanced across domains, which would add unnecessary complexity in clock tree synthesis.

Figure 11-7 on page 170 shows another example of the difficulty in managing scan chains across domains. Here a scan chain comes from power domain VCPU, which is a power-down domain, and connects to Scan Flop 2. Since this connection did not exist in the original design description, the path is not isolated and therefore additional isolation logic needs to be added. This may mean that we have to add more UPF commands.

## Recommendations:

- Try to implement a scan strategy that is power domain aware – minimize the number of times scan chain(s) cross power domain interfaces to restrict the number of isolation cells and/or level shifters.
- Scan chain reordering should also be power domain aware and where possible scan chains should be reordered within the power domain only.

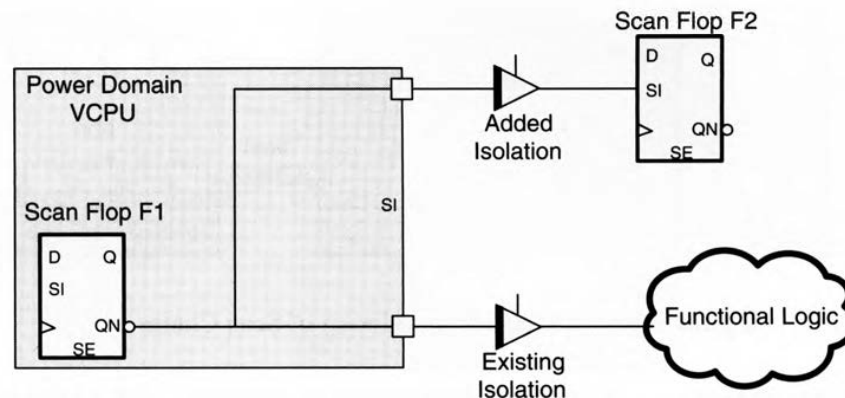


**Figure 11-7** Additional Isolation Needed For Scan Chain Creation

### 11.3.7 Always-On Network Synthesis

When implementing any power gating design, we need to make sure that the control signals from the power gating controller remain alive during shutdown. These control signals include the controls for the power switches, isolation cells, and retention registers.

The networks of buffers that distribute these control networks are referred to as always-on networks and must use only always-on buffers and inverters. These always-on cells are very similar to their standard counterparts but are connected to the always-on supply instead of the local switched supply. This ensures that the signals on these control networks remain alive during shutdown.

In most EDA solutions available today, identification of these always-on networks is fully automated within the synthesis process. Retention registers, isolation cells and power supply switches have specific pins annotated with always-on attributes that enable the synthesis of always-on networks. Similar attributes exist on specific buffers and inverters in the library that the synthesis tool can then select to buffer these networks.

A basic power gated supply with state retention register is shown in the diagram below. The control of the power gating switch (SLEEP) and the retention register control (RETN) are both buffered by always-on networks in the design. These pins are connected to the power gating controller through a network of always-on buffers and inverters that are connected to always-on power supply VDD. The local logic in the design is connected to the switched power supply VVDD delivered by the power gating switch.

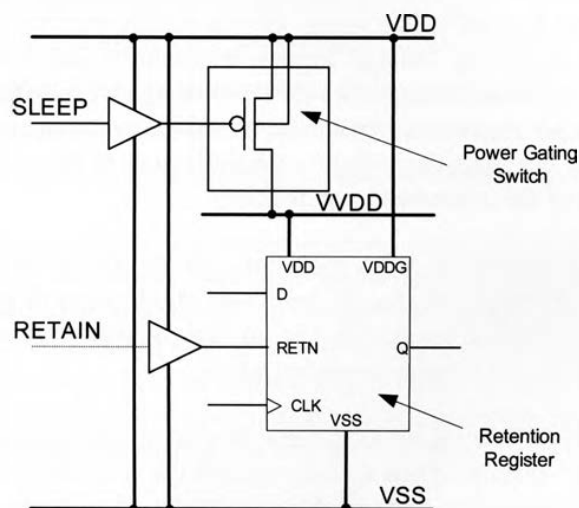


**Figure 11-8** Always-on Networks Required in Power Gated Designs

Although the synthesis and physical design tools understand always-on power routing, we still need to verify the always-on nets as part of Low Power Validation at the end of the design process. The design and specification of these networks is complex; they are usually inferred from the commands that specify the isolation and retention strategies. It is easy to make a mistake in these commands but sophisticated rule-based checkers can help spot any mistakes in this area.

## 11.4 Multi Corner Multi Mode Optimization with Voltage Scaling Designs

It is commonplace in modern SoC designs to have a variety of functional modes. With multi-voltage design we introduce a number of additional modes to the design, including various performance levels and sleep modes, such as light sleep, hibernate and complete shutdown. These additional modes need to operate at a number of different corners (process, voltage and temperature points). The combination of these

multiple modes each operating at different corners can be termed as scenarios. During the implementation process we need to both optimize and analyze the design for each of these scenarios.

For a dynamic voltage scaling design, we are concerned with ensuring full device operation at each of the various performance levels that are specified for the design. Specifically, we need to ensure that as we scale the voltage in conjunction with scaling the frequency that we have an optimal implementation at each supported combination of voltage and frequency. In a single voltage design, our worst case corner (for setup) is usually at the lowest voltage, highest temperature and worst case process. However, as we scale the performance levels in our design we may find that our worst case corner is some intermediate combination of voltage and frequency that does not reside at the edge of our operating window. Identification of this corner is important and an integral part of the implementation process.

This process is managed in the EDA tools through the use of multi-corner, multi-mode analysis. The designer can specify the modes of operation of the design and the corners at which the design is to operate. The implementation tools use these scenario specifications as constraint corners for optimization.

For the ARM1176JZF-S we intend to operate the design with varying voltage levels applied to the VCPU domain. When we vary the voltage at which VCPU operates, we also vary the clock frequency. Each of these modes, each of these combinations of voltage and frequency, has a number of associated timing corners representing some variation in voltage, temperature and process. In addition, each of these modes will have a set of associated timing constraints, unique to each corner, which the implementation of the design must satisfy.

In using multi-corner, multi-mode techniques we need to get the best balance between quality of results and the turn time for each pass of implementation. Specifying dozens of operating modes and corners will likely result in enormous run times for the implementation tools and will not necessarily yield the best quality of results. Specifying a few well considered modes and corners for implementation followed by a more comprehensive set of modes and corners for signoff is typically the best approach.

Each defined mode needs to be accompanied with a comprehensive set of constraints (SDC) that are used during both the implementation of the design as well as the signoff.

We specify the constraints for the restricted set of scenarios at synthesis and carry them through the entire implementation flow. We then use the constraints for the full set of scenarios for signoff static timing analysis.

## 11.5 Design Planning

The design planning process is critical to the implementation of a low power design. It is during design planning that we are presented with the majority of challenges and have the greatest opportunity to adversely affect the QoR of the design if we are not careful. Implementing a multi-voltage power gated design is going to impact the resulting QoR. Degradation of QoR can be caused by a number of factors including:

- Reduced operating voltage for certain parts of the logic
- Added delay introduced by level shifters and isolation cells
- Increased IR drop across switch networks in the power mesh
- Increased congestion caused by the use of switch cells and retention flops
- Placement restrictions due to physically bounded voltage areas

The physical design phase of the implementation starts with a determination of the overall topology of the design. Typically, the design topology will be governed by many factors, most of which are not necessarily low power related. The location of inputs and outputs for a design will be heavily influenced by the nature of the application into which the design is to be placed. Similarly, the location of the memory system and other blocks of hard IP within the design will dominate the topology, thereby reducing the flexibility and degrees of freedom available to the designer.

We need to overlay the power intent on the physical design process such that we do not limit the resulting QoR unnecessarily or place undue restrictions on the placement optimization process.

### 11.5.1 Creating Voltage Areas

During design planning, we will be physically bounding our power domain with voltage areas. This will ensure that all logic that constitutes a specific power domain is placed close together, allowing power to be delivered to that logic more efficiently.

When architecting voltage areas, we need to try to optimize the relative location of functional blocks and their communication with other blocks in the design. Where possible, we want to route power consuming busses efficiently and avoid unnecessary scenic routes around power domains. As with most design planning tasks, the creation of a voltage area from a power domain will be an iterative process and will involve identifying the best size and shape of the voltage area and how best to fit that voltage area in the chip.

We determine the size and shape of the voltage area based on the performance requirements of the logic and how densely it can be packed. Also, we consider the type of logic that is to be placed - does the voltage area contain a number of hard IP blocks that would present a significant challenge to the router if placed in the middle

of the design? Would using rectilinear voltage areas with these types of power domain help in the floorplanning process? We also need to make sure that voltage areas do not overlap.

Since voltage areas by definition contain logic that employ differing power strategies, care must be taken at the interface to these voltage areas to ensure that the voltage domains are truly isolated. We recommend that each voltage area be surrounded by a guard band providing some degree of physical isolation between the power domains. Guard bands present a physical placement constraint for the voltage area and they should be large enough to provide the necessary isolation but not so large that they use up valuable silicon real estate and impact the QoR. Figure 11-9 on page 174 shows how the voltage areas may be created.

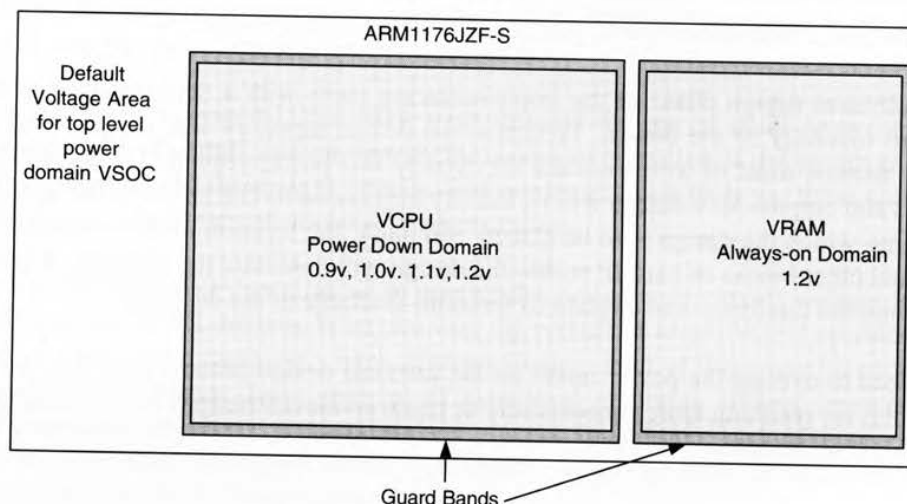


**Figure 11-9** Mapping of Power Domain to Voltage Area

As mentioned in Chapter 7, nesting of power domains is a logical concept and should be avoided in the physical design. If they are absolutely required, care must be taken to ensure that the physical bounds of the nested voltage areas are separate from the voltage area of the parent power domain.

We also do not recommend splitting a power domain into multiple separate physical regions on the chip. Although this is supported in most EDA tools today, having to do this should prompt the designer to reconsider the design partitioning.

Level shifters place specific constraints on design planning. For electrical integrity, the level shifters need to sit as close to the boundary of a voltage area as is possible, as they have input signal nets that are protected from buffering. Long un-buffered nets can create timing problems in a design.

When there are many signals between two power domains, the large number of level shifters required can present a challenge. For example, power domains that provide data on a number of wide data busses will naturally require large numbers of levels shifters, all of which will be pulled as close to the physical boundary of the voltage area as is possible. This can result in multiple site rows being consumed with level shifters around the periphery of a voltage area, leading to degradation in QoR for those nets that have their level shifter placed furthest from the boundary.

In these situations, it is recommended that power domains be floorplanned in such a way as to align data busses across power domain boundaries and give as large a dimension as possible to common interfaces. This will allow for spreading of the level shifters, resulting in a smaller physical separation between the logic in each power domain.

### 11.5.2 Power Gating Topologies

Where local power gating is employed, there are a number of alternative topologies for the placement of the power gating switches. These are discussed in detail in Chapter 14.

For most designs, including the ARM1176JZF-S, we use a grid design for the power network. With modern tools, we have two options for specifying the placement of the switches. We can specify an x and y increment, and the tool will place the switches evenly throughout the network. Or we can give the tool a design goal (such as current capability or IR drop) and the tool can do switch placement exploration and return with a list of options for the designer.

The size and placement of the switches is critical in determining the IR drop across the switch network. This in turn is critical in determining timing, since delay is very power-supply dependent.

In UPF, power gating switch cells are specified by the following command

```
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}}*
    {-on_state {state_name input_supply_port {boolean_function}}}*
    [-on_partial_state {state_name input_supply_port {boolean_function}}]*
    [-ack_port {port_name net_name [{boolean_function}]}]*
    [-ack_delay {port_name delay}]*
    [-off_state {state_name {boolean_function}}]*
    [-error_state {state_name {boolean_function}}]*
```

Where:

|                            |                                                                                                           |
|----------------------------|-----------------------------------------------------------------------------------------------------------|
| <i>switch_name</i>         | specifies the name of the switch instance to create;                                                      |
| <i>domain_name</i>         | specifies domain containing the switch.                                                                   |
| <b>-output_supply_port</b> | identifies the output supply port of the switch and the net where this port connects.                     |
| <b>-input_supply_port</b>  | identifies the input supply port of the switch and the net where this port connects.                      |
| <b>-control_port</b>       | identifies a control port on the switch and the net where this control port connects.                     |
| <b>-ack_port</b>           | identifies the acknowledge port on the switch and the signal net where this port connects.                |
| <b>-ack_delay</b>          | identifies the acknowledge port and delay on the switch where this port connects.                         |
| <b>-on_state</b>           | defines what value of the control port turns the switch on.                                               |
| <b>-on_partial_state</b>   | defines what value of the control port turns puts the switch in a current-limited state.                  |
| <b>-off_state</b>          | defines what value of the control port turns the switch off..                                             |
| <b>-error_state</b>        | identifies any error states, which if defined on the switch can be flagged during simulation or analysis. |

### 11.5.3 In-rush Current Management

The power-up sequence for a power gated design is critical for limiting the in-rush current, so that we do not cause voltage spikes that could corrupt retention registers or data in adjacent domains.

One effective technique, employed on ARM1176JZF-S, uses a buffered switch which provides a buffered (SLEEPOUT) version of the switch control input pin (SLEEP). This allows daisy chaining of the sleep control network throughout the design by connecting the SLEEP pin of one switch to the SLEEPOUT pin of another switch.

The exact topology of this daisy chain can be user defined, but the end goal is the same – a buffered sleep network that can be turned on quickly but at the same time limiting the in-rush current. This usually entails grouping sets of switches together, and driving the whole group with the same SLEEP pin, so they all turn on at the same time. One of the SLEEPOUT signals from the group is connected to the SLEEP pin of the next group. Thus, groups of pins are daisy chained. The speed vs. noise trade-off now involves determining the size of the group. This can be done through power analysis.

### 11.5.4 Recommendations:

- Voltage areas should be guard banded where necessary.
- Placement optimization must be constrained to ensure that cells in a power domain are placed inside the associated voltage area.

- New cells created during the physical design optimization process must be placed in the appropriate voltage area and powered by the correct power supply.
- Short feed-through paths in a voltage area should be protected from repeater insertion during design optimization and design-rule fixing to maintain wire connection.
- The isolation cells and level shifters should be placed at the voltage area boundary to ensure minimal impact to timing.
- It may be necessary to experiment with various power switch topologies and compare the results to determine which is most suitable for the design in question.
- Consider the power routing impacts for the switch topology chosen.

## 11.6 Power Planning

With a physical topology in place we can now begin to consider how to provide power to the various voltage areas in the design. Since by definition, each power domain employs a different power strategy, it is likely that we will be routing different power rails to each voltage area.

Minimizing the voltage drop across each of these power rails is a key part of meeting the performance goal. Unfortunately, many of the techniques that we employ in a low power design can make the voltage drop and noise problem worse. For example, when the design is operating at reduced voltage levels, our available margin for voltage drop is considerably reduced. When we change performance levels or power-up a block after shutdown, the on-chip current gradients become worse which can lead to noise injection. And in power gating blocks, there is an IR drop across the power switches.

Most EDA tools provide some level of automated power network synthesis (PNS) capability for the distribution of power across a design. We recommend highly that the implementation of a multi-voltage design utilize this type of PNS capability wherever possible. The number of concurrently changeable variables in a multi-voltage system is such that attempting to construct a power plan manually may result in a sub-optimal result both in terms of routing resources consumed and the voltage drop on the rails.

Power network synthesis allows the designer to specify the absolute constraints on the power plan (such as maximum voltage drop, routing layers and via requirements). It ensures that the power budget for the design is satisfied and provides integrated voltage drop and electro-migration analysis. Most level shifters available today that are shifting from a low voltage to a higher voltage require two power supplies. Cells that combine level shifting and isolation require a secondary supply that provides always-on power to the cell. During the power planning process it is necessary to hookup the

primary power pins to the appropriate voltage rails. There are various ways that this can be achieved depending on which EDA tools are being used. For example, for isolating level shifters, the primary power supply pins can be hooked up to the appropriate voltage rails through PNS and the secondary power pin (for always-on power) can be connected by the power router later in the design flow when the final placement of these cells is frozen.

The sleep transistor power network is composed of three components: a permanent power network, a virtual power network and an array of the sleep transistors. All of these components contribute to the quality of the sleep transistor power network in terms of IR drop, routing resources and silicon area. Consequently, the synthesis of the sleep transistor power network becomes a challenge. Historically, most power-gating designs used scripts to insert and place the switch cells, and create power straps and rails based on heuristic rules. Today, the sleep transistor power network can be synthesized using a power planning tool PNS [PNS] that considers all the three components.

#### Recommendations:

- Use a power planning tool which supports switch cells to create the sleep transistor power network including switch cell insertion and placement.
- Run both static and dynamic IR drop analysis on a design with switch cells to verify the power integrity. Significant static IR drop violations usually require increasing the number of switch cells and adjusting their positions.
- Dynamic IR drop violations are commonly fixed by decoupling capacitor insertion. Accurate power-on sequence I-V responses have to be obtained by transient analysis to assure that all IR drop violations are fixed.
- In the case of any IR drop violations, try to fix the violations in the permanent power network, because any changes in the switch cells distribution and virtual power network usually require re-analyzing wakeup latency and peak current.
- Run wakeup in-rush current analysis to check that the max in-rush current and wakeup latency meet the defined constraints. For small designs, it is possible to run this analysis in SPICE. At chip level, the wakeup analysis is done using rail analysis tools.
- Run dynamic IR drop analysis covering the wakeup period to detect any violations caused by current spikes, through common power rails, on the parts of the design that are alive while the power-gated block is waking up.

### 11.6.1 Decoupling Capacitor Insertion

An integral part of the power planning process involves ensuring that the noise coupling between different power supply rails and between cells in the same power domain is minimized. This power and ground noise can degrade timing and possibly cause functional failures. Inserting decoupling capacitors (decap cells) in the design is the method commonly used to address these issues.

Decoupling capacitor insertion becomes more challenging in power-gating designs than in normal designs. Implementing in-rush current control techniques, such as daisy chaining the power-gating structure to sequentially charge the power-gated block, reduces the in-rush current. However, the turn-on current of each switch, particularly of those switches which are close to the head of the daisy chain and turned on first, can still generate significant power and ground noise pulses in the rest of the design, through the permanent power supply and common ground networks. Adding decoupling capacitors to the permanent power supply nets is an effective way to reduce this noise. It is most effective to insert the decoupling capacitors next to the switch cells which are the sources of the noise during wakeup.

Apart from the power and ground noise generated by the in-rush current, there is also the noise generated by cell switching currents in the power-gated block during normal operation. Decap insertion is a common method to address this noise. However, decap insertion becomes more challenging in the power-gated design. In sub-90nm technology, decap cells exhibit high leakage power that needs to be controlled. Also, the decap cells added to the virtual power network have significant impact on the wakeup latency and in-rush current. The larger the decoupling capacitance, the larger the in-rush charge current and the longer the charge time (latency) at wakeup. Therefore, the optimization of de-cap insertion in the power-gating design becomes very important to achieve maximum noise reduction with minimum added capacitance at the virtual power network. This can be done by identifying noise hot spots using dynamic IR drop analysis tools and then inserting just enough capacitance at the hot spots to reduce the noises meeting defined noise target. It is worth noting that advanced EDA tools have implemented such optimal decap insertion methods.

#### Recommendations:

- Add as much decoupling capacitance as permitted in the permanent power network at the positions close to the switch cells. This achieves the maximum effectiveness and minimum impact on the wakeup latency and in-rush current. It is convenient to integrate the decap into the switch cell to simplify decap insertion. The maximum capacitance of the decap on the permanent power network is constrained by the leakage and area penalties.

- To fix dynamic IR drop violations in the post-layout stage, it is preferable to add decoupling capacitance to the permanent power network close to the violation spots, if the violations are related to the permanent power network. The rest of the violations have to be fixed by adding decap to the virtual power network at the violation spots.

## 11.7 Clock Tree Synthesis

The impact on the overall power consumption of the clock tree in a design is significant. In many cases, more than half of the overall power consumption of a design may be due to the clock network. There are a number of ways to mitigate the effects of the clock tree power, including clock gating and minimizing clock tree insertion delays.

However, when we are dealing with a multi-voltage design, we have additional restrictions on how we can manipulate the clock tree to meet both our low power requirements and our performance requirements at each performance level. In a single clock multi-voltage design, the clock is used in multiple power domains and hence crosses a number of voltage area boundaries. As the clock passes through each voltage area its latency is modified based on the voltage at which those voltage areas are operating. If the clock path buffering and data path buffering are not well balanced across voltage areas then skew management becomes very difficult.

The diagram in Figure 11-10 illustrates the issue. In this particular situation, both the data and the clock are being sent by the CPU to the memory. The data path buffering is split between the VCPU and VRAM power domains with the clock path buffering residing primarily in the VRAM power domain. Assume that we have very good skew minimization when both power domain VCPU and VRAM are operating at the same supply voltage. When the voltage on power domain VCPU is reduced as part of performance scaling then we run the risk of slowing the data path relative to the clock and introducing setup violations in power domain VRAM.

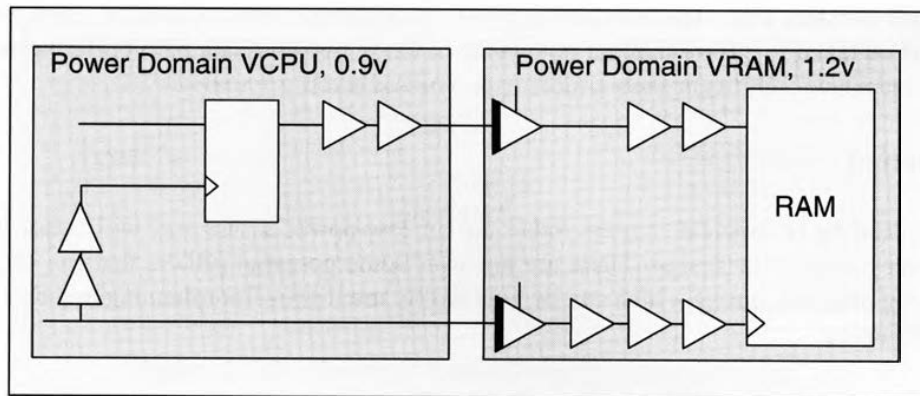


**Figure 11-10** Clock Tree Synthesis Challenge in a Multi-Voltage Design

In order to help alleviate this problem, clock tree synthesis algorithms in current EDA tools are multi-voltage aware and use a bottom-up approach to constructing the clock tree. Each voltage area is processed in turn and the clock networks for each voltage area are constructed to minimize the skew. These low level clock tree structures are then merged by the construction of higher level clock trees that join together the sub-tree to form an overall clock network for the design.

Adopting this approach helps locate the buffering for the clock tree in the same voltage area as the data path buffering. This clock buffer clustering prevents the clock network from winding its way in and out of a number of voltage areas, requiring level shifters to be inserted every time the clock network crosses a voltage area boundary. This level shifter insertion process would add significant delay to the clock tree and impact the ability to minimize the clock skew and power. Figure 11-11 illustrates this clustering approach. Note that the VRAM and VCPU domains have level shifters ("LS" in the figure) in their clock trees.

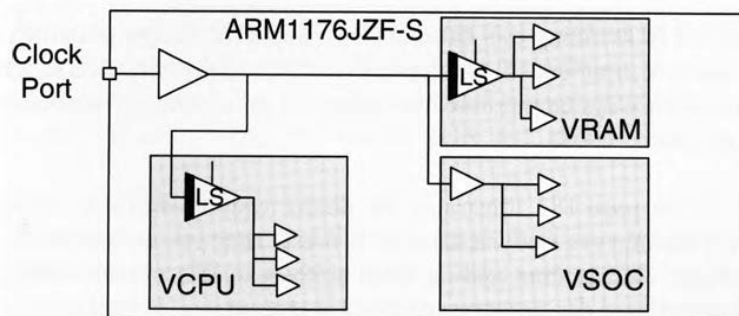


**Figure 11-11** Bottom Up Clock Tree Clustering

This approach lends itself well to voltage areas that have a good mix of clock and data buffering such as blocks of standard logic. However, when we have a voltage area

which contains little standard cell logic but rather many blocks of hard IP, extra care must be taken to ensure that the imbalance of clock tree and data path buffering will not introduce significant skew problems as voltage levels are scaled.

## Routing

The leading commercial routers today are all low-power aware and will honor the power intent in the design. There are however some potential pitfalls that can occur during detailed routing which can be avoided by intelligent floorplanning and design partitioning.

When a design is partitioned into multiple voltage areas, hard placement and routing restrictions exist that can impact the routing of the design. This can lead to degradation of QoR. It is therefore important to consider routing of the design, especially on-chip bus structures and global control networks, when floorplanning the design.

If the best route for a net from one part of the design to another is across one or many voltage areas then a decision has to be made on how to deal with the net.

- Route the net through the voltage area(s) and then add the appropriate level shifters to each voltage area crossing
- Detour the route around the voltage area(s) to the destination

With the first option, adding level shifters to every point at which the net crosses a voltage area will be expensive in terms of area, power and delay. The second option will introduce an incremental delay compared to a direct connection that would be available in the single voltage design; however the impact of this approach is considerably less than with the first option.

Both of these options will impact the QoR and so clearly we want to avoid this situation. The cost (in time) of rectifying this situation at the detailed routing stage will be expensive and so avoiding this situation by considered design planning earlier in the design cycle is recommended. Complete avoidance of this situation in a multi-voltage design may not always be possible but making every attempt to minimize the problem is recommended.

Figure 11-12 on page 183 illustrates the detour routing problem. Here a net going from power domain A to power domain C must cross power domain B which is at a higher voltage. Rather than adding level shifters to this net and routing it through power domain C, we can detour route this net around power domain C.

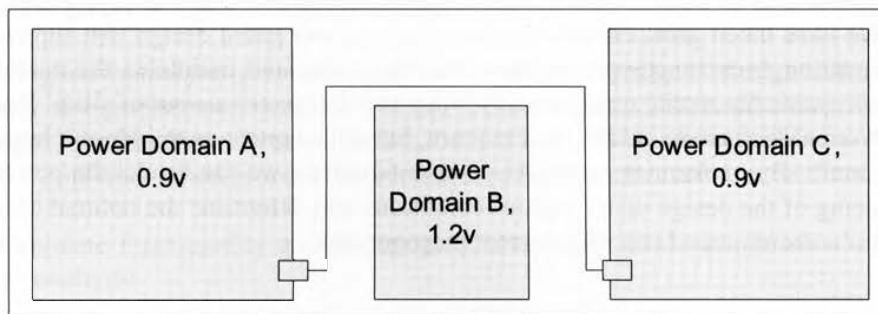


**Figure 11-12** Detour Routing Around a Power Domain

## 11.8 Power Analysis

Having completed the implementation of our multi-voltage, power gated design, it is now necessary to verify the integrity of the power network. Two aspects of our power network are critical: the voltage drop seen by the standard cells in the power gated blocks and the profile of the in-rush current during power-up sequencing.

Performing power rail analysis of the design is critical. There will be many power rails in the design and the integrity of each needs verifying and the interaction between rails (coupling) also needs to be verified. Using a comprehensive rail analysis tool, we can perform extensive analysis of the power networks in the design. The coarse level power network analysis was completed during the power planning phase. What we are looking for here are areas in the design where we have excessive voltage drop due to local level clustering which results in power “hot” spots.

Implementation tools today are very good at performing local clustering of registers, clock gates and clock tree buffers to ensure that aggressive timing goals are met. This however can make the voltage drop problem worse in those regions of the chip, especially in regions with switched power rails. Since logic spreading in these areas is not viable (this will impact the QoR) we must deal with the voltage drop problem in a different manner and typically sizing the switch cells and power mesh is the best approach.

The rail analysis gives us valuable data on the voltage drop seen by every standard cell in the design. Where we have an unacceptably large voltage drop we can try sizing or duplicating power switch cells and sizing the power mesh to alleviate the problem. Creation of instance specific voltage drop data for a design can then be used by a signoff static timing to measure the impact of voltage drop on the timing of the design.

The second area of power analysis that needs to be addressed is that of in-rush current. We have taken great effort in architecting a power gated design that supports a fast transition from functional mode to standby mode, and maintains the necessary state to resume functional mode quickly when the design comes out of sleep. Recovering from a sleep mode quickly is important, but not so quick as to induce a large in-rush current. By performing extensive transient analysis we can model the power-up sequencing of the design under various conditions and determine the optimal balance between wake-up speed and rush current management.

## 11.9 Timing Analysis

A multi-voltage power gated design will need to function in a number of different modes each operating at different corners. During implementation, we selected a reduced set of scenarios - operating modes and corners - and used these as constraints. Hopefully we chose the best and worst cases and produced a design that will work under all scenarios.

The purpose of signoff static timing analysis is to prove that we have, in fact, implemented a design that works under all scenarios. As such it is necessary to accurately specify all of these scenarios and provide all the necessary technology libraries required to analyze the design at the supported performance levels.

In particular, we need to make sure that the libraries - especially level shifters - are characterized for the voltages as which we run signoff static timing analysis.

Run times will now be significantly longer than for a design that is not implemented with aggressive low power techniques.

During the fabrication process for a CMOS based design, it is expected that there will be minor variations in resistance and capacitance of both the transistors and the metalization. These variations will impact the timing paths in different regions of the chip. We recommend that a small window of uncertainty be placed around each of these operating corners to provide a greater level of confidence in the final signoff results. This window of uncertainty can help counter any on-chip variation (OCV) that may exist across the die. On-chip variation will derate the path delays in unfavorable directions, such as speeding up clock and slowing down data for setup checks and visa-versa for hold checks.

## 11.10 Low Power Validation

Having successfully implemented our multi-voltage power gated design that meets all power and performance targets, it is now necessary to validate the integrity of the design. Specifically we want to ensure that the low power intent provided at the start of the implementation process has been successfully implemented in the final design.

We validate functionality and the integrity of our low power implementation using three methods:

- Gate level logic simulation
- Equivalence checking
- Rule-based methods

Clearly gate level simulation of our final design can tell us if the design still remains functional with the low power structures. Specifically, we can validate that the design:

- resets cleanly at startup
- can be placed into various sleep modes
- behaves appropriately during shutdown
- powers-up successfully after shutdown

Formal equivalence checking tools can prove that the gate level netlist is equivalent to the original RTL plus UPF code.

A rule-base tool can tell us that the power structure of the final gate level netlist makes sense. It can validate that the isolation cells and level shifters are placed in the correct domains, and that all nets requiring isolation or level shifting have the appropriate cells in place. It can check that cells that required always on power – such as retention registers, isolation cells, and buffers of power control signals – do, in fact, have the appropriate supplies. Finally, we can use these tools to find situations where there are redundant isolation cells or level shifters.

## 11.11 Manufacturing Test

In a typical voltage scaled system, various parts of the design will be running at reduced power supply levels. During the implementation process, the design can be optimized at these voltage levels thereby yielding an overall lower power design that meets the needs of the system application. However, in most manufacturing test situations, the design will be tested at nominal supply rail voltage levels which will cause the design to consume significantly more power during test than it would during nor-

mal operation. If the design has only been packaged assuming functional power dissipation numbers then thermal failure of both the design and package at test can result. In addition, this increased power consumption in the core can lead to significant voltage drop in the power distribution network which in turn leads to functional failure. It is therefore necessary to consider the requirements of both the functional and test modes of the design when implementing voltage scaling.

When performing manufacturing test on designs implemented in very small geometries (90/65/45 nm) it is necessary to use a variety of test models and test strategies. Specifically, testing the design with stuck-at faults models now needs to be complemented with transition fault delay testing. Delay fault testing is particularly susceptible to changes in the voltage and frequency at which a design operates. Similarly, path based testing is also dependant upon the voltage at which the design is being operated. As will be discussed later in this chapter, the critical path of a voltage scaled design is not the same path at each voltage level at which the design operates. Each performance level supported by the design may well have a unique critical path. Each of these paths needs to be tested during at-speed test in a manufacturing test environment.

However, as we reduce the supply voltages at which we test our design, delay behavior changes dramatically. In particular, weak circuit elements and defective circuit elements do not behave in the same way. In addition, the tester environment differs from the system environment in noise content, heat dissipation and pin load. As a result, care needs to be taken to distinguish between correct behavior and defective behavior. In many cases, final decisions on correct timing and voltage levels will need to wait until silicon is available.

Automating the manufacturing test process will require performing stuck-at fault testing and delay fault testing at several operating voltages, and also may involve both nominal and high temperature test. Developing the appropriate mix of tests in order to guarantee sufficient quality at reasonable cost is a complex problem. Different trade-offs between test time and coverage will apply to different products.

This chapter describes the syntax for selected UPF commands referenced in the text.

Excerpts from the “Unified Power Format(UPF) Standard, Version 1.0” reproduced by permission of Accellera. Copyright(c) 2006-2007 by Accellera. Accellera does not warrant or represent the accuracy or content of the excerpted material, and expressly disclaims any express or implied warranty. Accellera Standards excerpts are supplied “AS IS.”

The full standards document in its entirety can be found under [www.accellera.org](http://www.accellera.org).

## B.1 add\_pst\_state

|                     |                                                                                                                                                               |
|---------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Purpose</b>      | Define the states of each of the supply nets for one possible state of the design                                                                             |
| <b>Syntax</b>       | <b>add_pst_state</b> <i>state_name</i><br>-pst <i>table_name</i><br>-state <i>supply_states</i>                                                               |
| <b>Arguments</b>    | <i>state_name</i> The power state.                                                                                                                            |
|                     | -pst <i>table_name</i> The power state table (PST) to which this state applies.                                                                               |
|                     | -state <i>supply_states</i> The list of supply net state names , listed in the corresponding order of the -supplies listing in the <b>create_pst</b> command. |
| <b>Return value</b> | Return a 1 if successful or a 0 if not.                                                                                                                       |

The **add\_pst\_state** command defines the states of each of the supply nets for one possible state of the design.

It is an error if the number of *supply\_state\_names* is different than the number of supply nets within the PST.

Syntax example:

```
create_pst          pt -supplies { PN1   PN2   SOC/OTC/PN3 }
add_pst_state s1 -pst pt -state { s08   s08   s08      }
add_pst_state s2 -pst pt -state { s08   s08   off       }
add_pst_state s3 -pst pt -state { s08   s09   off       }
```

## B.2 connect\_supply\_net

|                     |                                                                                                                                                                                                                                                 |
|---------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Purpose</b>      | Connect a supply net to supply ports and/or pins                                                                                                                                                                                                |
| <b>Syntax</b>       | <b>connect_supply_net</b> <i>net_name</i><br>[-ports <i>list</i> ] [-pins <i>list</i> ]<br>[<-cells <i>list</i>   -domain <i>domain_name</i> >]<br>[<-rail_connection <i>rail_type</i>   -pg_type <i>pg_type</i> >]*<br>[-vct <i>vct_name</i> ] |
| <b>Arguments</b>    | <i>net_name</i> The name of the supply net.                                                                                                                                                                                                     |
|                     | <b>-ports</b> <i>list</i> A list of ports to connect.                                                                                                                                                                                           |
|                     | <b>-pins</b> <i>list</i> A list of pins on cells to connect.                                                                                                                                                                                    |
|                     | <b>-cells</b> <i>list</i> A list of cells to use for <b>-rail_connection</b> or <b>-pg_type</b> .                                                                                                                                               |
|                     | <b>-domain</b> <i>domain_name</i> The domain to use for <b>-rail_connection</b> or <b>-pg_type</b> .                                                                                                                                            |
|                     | <b>-rail_connection</b> <i>rail_type</i> The rail type (for older libraries).                                                                                                                                                                   |
|                     | <b>-pg_type</b> <i>pg_type</i> The power/ground pin type.                                                                                                                                                                                       |
|                     | <b>-vct</b> <i>vct_name</i> A VCT defining how values are mapped from UPF to an HDL model or from the HDL model to UPF.                                                                                                                         |
| <b>Return value</b> | Return the fully qualified name of the supply net if successful or a 0 if not.                                                                                                                                                                  |

Any ports, pins, cells, supply nets, or domains are referenced relative to the current UPF scope .

The **connect\_supply\_net** command provides an explicit connection of a supply net to any port and overrides (has higher precedence than) the auto-connection semantics that might otherwise apply. **-domain** or **-cells** is required when the **-rail\_connection** or **-pg\_type** options are specified.

Use:

- ports to connect to supply ports;
- pins to connect to pins on library cells;
- cells to connect to all pins of the appropriate type (power or ground) on the specified cells;

**-rail\_connection** to connect to pins having this (rail) type; only use this if the **-cells** or **-domain** options are specified;

**-pg\_type** to connect only to pins having this power/ground type (the pin type defined in the library model is used for determining the connection); only use this if the **-cells** or **-domain** options are specified;

**-vct** to indicate for every HDL port to which the port is connected, the supply net state shall be converted if it is being propagated into the HDL port, or the HDL port value shall be converted if it is being propagated onto the supply net. **-vct** is ignored for any connections of the supply net to pins or supply ports defined in UPF.

The following also apply.

- The **-ports** and **-pins** options are mutually exclusive with the **-cells**, **-domain**, **-rail\_connection**, and **-pg\_type** options.
- The **-rail\_connection** and **-pg\_type** options are mutually exclusive with each other.
- A supply net can only be connected to a port, pin, or cell in the same *extent* as the domain of that supply net.
- It is an error if *net\_name* has not been previously created; in this case, a 0 shall be returned.
- It is an error if any design element specified in this command does not exist.
- It is an error if the value conversions specified in the VCT do not match the type of the HDL port.

Syntax examples:

```
connect_supply_net v09 -ports {VDD U18/v9 U21/v9}

connect_supply_net pd1_vdd
  -ports pll_inst/vdd
  -vct upf2vlog_vdd
```

### B.3 create\_power\_domain

|                     |                                                                                                                                             |                                                                       |
|---------------------|---------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------|
| <b>Purpose</b>      | Define a power supply distribution network for a set of design elements                                                                     |                                                                       |
| <b>Syntax</b>       | <b>create_power_domain</b> <i>domain_name</i><br>[-elements <i>list</i> ]<br>[-include_scope]<br>[-scope <i>instance_name</i> ]             |                                                                       |
| <b>Arguments</b>    | <i>domain_name</i>                                                                                                                          | The new power domain; this shall be a simple (non-hierarchical) name. |
|                     | -elements <i>list</i>                                                                                                                       | Use this set of design elements in the power domain.                  |
|                     | -include_scope                                                                                                                              | Include the scope of the domain in the extent of the power domain.    |
|                     | -scope <i>instance_name</i>                                                                                                                 | Create the power domain within this logic hierarchy scope.            |
| <b>Return value</b> | Return the fully qualified name of the domain (from the current scope) that is created or a null string if the power domain is not created. |                                                                       |

The **create\_power\_domain** command defines a power supply distribution network, usually for a list of design elements in the power domain. A *power domain* is a logical grouping of one or more design elements. A power domain shall have one primary power net and one primary ground net. A power domain may have additional supply nets, supply ports, and/or switches.

A power domain is functional only if the primary power and ground supply nets are specified.

**-elements** (or the **add\_domain\_elements** command) specifies the complete set of design elements contained within the power domain, i.e., those instances and all their children (unless specified otherwise).

- *list* is specified relative to the current scope; it is not influenced by the **-scope** argument.
- If **-include\_scope** is also specified, the scope of the domain is included in the extent of the domain.
- When neither **-elements** nor **-include\_scope** is specified, the power domain consists of the current scope and any of its children not specified (as elements) in another **create\_power\_domain** command.

**-scope** specifies the scope, i.e., the instance, where the domain shall be created. This scope is the current scope for this command; it defines the domain boundary within the logic design. If **-scope** is not specified, the power domain shall be created within the current scope.

Syntax examples:

```
create_power_domain PD1 -elements {top/U1}  
  
set_scope /top/U1  
create_power_domain PD2
```

## B.4 create\_power\_switch

|                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
|------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Purpose</b>   | Define a switch in the power domain                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| <b>Syntax</b>    | <pre> <b>create_power_switch</b> <i>switch_name</i>   -<b>domain</b> <i>domain_name</i>   -<b>output_supply_port</b> {<i>port_name</i> <i>supply_net_name</i>}   {-<b>input_supply_port</b> {<i>port_name</i> <i>supply_net_name</i>}}*   {-<b>control_port</b> {<i>port_name</i> <i>net_name</i>}}*   {-<b>on_state</b> {<i>state_name</i> <i>input_supply_port</i> {<i>boolean_function</i>}}}*   [-<b>on_partial_state</b> {<i>state_name</i> <i>input_supply_port</i> {<i>boolean_function</i>}}]*   [-<b>ack_port</b> {<i>port_name</i> <i>net_name</i> [{<i>boolean_function</i>}}]*   [-<b>ack_delay</b> {<i>port_name</i> <i>delay</i>}]*   [-<b>off_state</b> {<i>state_name</i> {<i>boolean_function</i>}}]*   [-<b>error_state</b> {<i>state_name</i> {<i>boolean_function</i>}}]*           </pre> |
| <b>Arguments</b> | <i>switch_name</i> The name of the switch instance to create; this shall be a simple name.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
|                  | <b>-domain</b> <i>domain_name</i> The domain containing the switch.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
|                  | <b>-output_supply_port</b> <i>port_name</i> <i>supply_net_name</i> The output supply port of the switch and the net where this port connects.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
|                  | <b>-input_supply_port</b> <i>port_name</i> <i>supply_net_name</i> The input supply port of the switch and the net where this port is connected.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
|                  | <b>-control_port</b> { <i>port_name</i> <i>net_name</i> }                      A control port on the switch and the net where this control port connects.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
|                  | <b>-on_state</b> { <i>state_name</i> <i>input_supply_port</i> { <i>boolean_function</i> }}            A named state, the <i>input_supply_port</i> for which this is defined, and its corresponding Boolean function.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
|                  | <b>-on_partial_state</b> { <i>state_name</i> <i>input_supply_port</i> { <i>boolean_function</i> }}            A named state, the <i>input_supply_port</i> for which this is defined, and its corresponding Boolean function.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
|                  | <b>ack_port</b> { <i>port_name</i> <i>net_name</i> [{ <i>boolean_function</i> }}]            A named state, the <i>input_supply_port</i> for which this is defined, and its corresponding Boolean function where the switch is in a current-limited state.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
|                  | <b>-ack_delay</b> { <i>port_name</i> <i>delay</i> }                      The acknowledge port on the switch and the signal net where this port connects. Optionally, a boolean function can also be specified.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
|                  | <b>-off_state</b> { <i>state_name</i> { <i>boolean_function</i> }}            The acknowledge port and delay on the switch where this port connects.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |

|                     |                                                                                        |                                                                                                |
|---------------------|----------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------|
|                     | <b>-error_state</b><br>{state_name<br>{boolean_function}}                              | Any error states, which if defined on the switch can be flagged during simulation or analysis. |
| <b>Return value</b> | Return the full path name of the created switch if successful or a null string if not. |                                                                                                |

The **create\_power\_switch** command defines an instance of a power switch in the power domain. The switch is created within the scope of the power domain. The switch is either on or off.

The switch is on if the value on the control port(s) equals an “on” state Boolean expression; this drives the output port to the “on” state. If a value is specified on the input supply port, then that value is driven on the output supply port.

If the switch is not on, it is off, and an “off” state is driven onto the output port. Some “off” states can be identified as error states. The simulation semantics for these error states is tool dependent.

If a *boolean\_function* is specified for **-ack\_port**, the result of *boolean\_function* is driven on **-ack\_port**’s *port\_name delay* time units after a control port transition. Otherwise, a logic 1 shall be driven on the *port\_name delay* time units after the switch is closed and a logic 0 shall be driven *delay* time units after the switch is opened. *delay* (default is 0) may be specified as a unit-less natural integer or as a Verilog time unit. If specified as a natural integer, the time unit shall be the same as the simulation precision.

Any **-ack\_port**, **-on\_state**, or **-error\_state** *boolean\_functions* shall be SystemVerilog Boolean expressions.

The following also apply.

- It is an error if a specified control port does not exist. The existing connectivity of the control ports is unmodified.
- All states not covered by three states (on, off, and error) are anonymous error states.
- Each state name shall be unique.
- Any conflicting state definition is an error.
- No synthesis semantics are associated with any Boolean function, except the mapped switch has to provide that functionality.

Syntax example:

```

create_power_switch sw1
-domain PD_SODIUM
-output_supply_port {vout VN3}
-input_supply_port {vin1 VN1}
-input_supply_port {vin2 VN2}
-control_port {ctrl_small ON1}
-control_port {ctrl_large ON2}
-control_port {ss SUPPLY_SELECT}
-on_state {partial_s1 vin1 {ctrl_small & !ctrl_large & ss}}
-on_state {full_s1 vin1 {ctrl_small & ctrl_large & ss}}
-on_state {partial_s2 vin2 {ctrl_small & !ctrl_large & !ss}}
-on_state {full_s2 vin2 {ctrl_small & ctrl_large & !ss}}
-error_state {no_small {!ctrl_small & ctrl_large}}

```

## B.5 create\_pst

|                     |                                                                                      |                                                                                |
|---------------------|--------------------------------------------------------------------------------------|--------------------------------------------------------------------------------|
| <b>Purpose</b>      | Create a power state table with a specific ordering of supply nets                   |                                                                                |
| <b>Syntax</b>       | <b>create_pst</b> <i>table_name</i><br><b>-supplies</b> <i>list</i>                  |                                                                                |
| <b>Arguments</b>    | <i>table_name</i>                                                                    | The power state table name (PST).                                              |
|                     | <b>-supplies</b> <i>list</i>                                                         | The list of supply nets or ports to include in each power state of the design. |
| <b>Return value</b> | Return the name of the power state table if it is created or the null string if not. |                                                                                |

The **create\_pst** command creates a PST, using a specific order of supply nets.

A *power state table* is used for implementation — specifically for synthesis, analysis, and optimization. It defines the legal combinations of states, i.e., those combinations of states that can exist at the same time during operation of the design.

The power state table has no simulation semantics. It is tool-dependent whether simulation tools report an error if an illegal (unspecified) combination of states occurs.

It is an error if a specified supply net has not already been created.

Syntax example:

```
create_pst MyPowerStateTable -supplies {PN1 PN2 SOC/OTC/PN3}
```

## B.6 create\_supply\_net

|                     |                                                                                                                                        |                                                                                                                                                                                                   |
|---------------------|----------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Purpose</b>      | Create a power or ground supply net                                                                                                    |                                                                                                                                                                                                   |
| <b>Syntax</b>       | <b>create_supply_net</b> <i>net_name</i><br><b>-domain</b> <i>domain_name</i><br>[-reuse] [-resolve <unresolved   one_hot   parallel>] |                                                                                                                                                                                                   |
| <b>Arguments</b>    | <i>net_name</i>                                                                                                                        | The name of the supply net; this shall be a simple (non-hierarchical) name.                                                                                                                       |
|                     | <b>-domain</b> <i>domain_name</i>                                                                                                      | The domain in whose scope the supply net is to be created.                                                                                                                                        |
|                     | <b>-reuse</b>                                                                                                                          | Extend <i>net_name</i> as a supply net within <i>domain_name</i> . No new nets are created.                                                                                                       |
|                     | <b>-resolve</b> <unresolved   one_hot   parallel>                                                                                      | A resolution mechanism which determines the state and voltage of the supply net from the state and voltage values supplied by each of the individual switches. The default is <b>unresolved</b> . |
| <b>Return value</b> | Return the fully qualified name (from the current scope) of the created net or a null string if the net is not created.                |                                                                                                                                                                                                   |

The **create\_supply\_net** command creates a supply net. The net is defined for the power domain, created in the logic hierarchy at the same scope as *domain\_name*, and propagated through implicitly created ports and nets through the logic hierarchy as required.

The following also apply.

- It is an error if *domain\_name* does not indicate a previously created power domain.
- When **-reuse** is specified, it is an error if *net\_name* does not already exist.
- When **-resolve unresolved** is specified, this supply net only allows a single driver.

Syntax example:

```
create_supply_net v09
  -domain PD1
```

## B.7 create\_supply\_port

|                     |                                                                                                                           |                                                                                                       |
|---------------------|---------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------|
| <b>Purpose</b>      | Create a port on a power domain                                                                                           |                                                                                                       |
| <b>Syntax</b>       | <b>create_supply_port</b> <i>port_name</i><br>[-domain <i>domain_name</i> ]<br>[-direction <in   out>]                    |                                                                                                       |
| <b>Arguments</b>    | <i>port_name</i>                                                                                                          | The name of the supply port. Hierarchical names are allowed, unless <b>-domain</b> is also specified. |
|                     | <b>-domain</b> <i>domain_name</i>                                                                                         | The domain where this port defines a supply net connection point.                                     |
|                     | <b>-direction</b> <in   out>                                                                                              | The direction of the port. The default is <b>in</b> .                                                 |
| <b>Return value</b> | Return the fully qualified name (from the current scope) of the created port or a null string if the port is not created. |                                                                                                       |

The **create\_supply\_port** command defines a supply port at the scope of the power domain when **-domain** is specified, or at the current scope if **-domain** is not specified.

**-direction** defines how state information is propagated through the supply network as it is connected to the port. If the port is an input port, the state information of the external supply net connected to the port shall be propagated into the domain. Likewise, for an output port, the state information of the internal supply net connected to the port shall be propagated outside of the domain.

It is an error if *port\_name* creates a name conflict in the logic hierarchy or specifies a previously created port.

It is an error if *domain\_name* does not indicate a previously created power domain.

Syntax example:

```
create_supply_port VN1
  -domain PD1
  reg/wire, Bit, Logic
```

## B.8 set\_domain\_supply\_net

|                     |                                                                                                                                                            |
|---------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Purpose</b>      | Set the default power and ground supply nets for a power domain                                                                                            |
| <b>Syntax</b>       | <b>set_domain_supply_net</b> <i>domain_name</i><br>- <b>primary_power_net</b> <i>supply_net_name</i><br>- <b>primary_ground_net</b> <i>supply_net_name</i> |
| <b>Arguments</b>    | <i>domain_name</i> The domain where the default supply nets are to applied.                                                                                |
|                     | - <b>primary_power_net</b> The primary power supply net.<br><i>supply_net_name</i>                                                                         |
|                     | - <b>primary_ground_net</b> The primary ground net.<br><i>supply_net_name</i>                                                                              |
| <b>Return value</b> | Return 1 if it succeeds and 0 if it fails.                                                                                                                 |

The **set\_domain\_supply\_net** command associates the default power and ground supply nets to the logic elements of the power domain.

The *primary power and ground nets* are the default nets connected to the logic elements (or inferred cells) of the power domain. These power and ground nets are used for all the elements in the power domain. At the gate level, this means all power/ground pins of all the inferred gates are connected to the primary power/ground nets, unless specified otherwise in a **connect\_supply\_net**, **set\_retention**, or **set\_isolation** command.

It is an error if *domain\_name* does not indicate a previously created power domain.

It is an error if *domain\_name* already contains a primary power and ground net supply.

Syntax example:

```
set_domain_supply_net PD1
-primary_power_net PG1
-primary_ground_net PG0
```

## B.9 set\_isolation

|                                                                                                                                                                                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |                       |                          |                                   |                                                |                                                                                                                                                                                  |                                                                                                                                                                   |                              |                                                                                                             |                                         |                                                                            |                                              |                                                                                                       |
|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------|--------------------------|-----------------------------------|------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------|-------------------------------------------------------------------------------------------------------------|-----------------------------------------|----------------------------------------------------------------------------|----------------------------------------------|-------------------------------------------------------------------------------------------------------|
| <b>Purpose</b>                                                                                                                                                                   | Specify the elements in the domain to isolate using the specified strategy                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |                       |                          |                                   |                                                |                                                                                                                                                                                  |                                                                                                                                                                   |                              |                                                                                                             |                                         |                                                                            |                                              |                                                                                                       |
| <b>Syntax</b>                                                                                                                                                                    | <pre>set_isolation isolation_name -domain domain_name &lt;-isolation_power_net net_name   -isolation_ground_net net_name   -isolation_power_net net_name -isolation_ground_net net_name   - no_isolation&gt; [-elements list] [-clamp_value &lt;0   1   latch   Z&gt;] [-applies_to &lt;inputs   outputs   both&gt;]</pre>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |                       |                          |                                   |                                                |                                                                                                                                                                                  |                                                                                                                                                                   |                              |                                                                                                             |                                         |                                                                            |                                              |                                                                                                       |
| <b>Arguments</b>                                                                                                                                                                 | <table> <tr> <td><i>isolation_name</i></td><td>Isolation strategy name.</td></tr> <tr> <td><b>-domain</b> <i>domain_name</i></td><td>The domain for which this strategy is applied.</td></tr> <tr> <td> <b>&lt;-isolation_power_net net_name  <br/> -isolation_ground_net net_name  <br/> -isolation_power_net net_name<br/> -isolation_ground_net net_name  <br/> -no_isolation&gt;</b> </td><td>The supply net(s) used to supply the isolation logic inferred by this strategy. Does not isolate the port, pin, or design element specified in the elements list.</td></tr> <tr> <td><b>-elements</b> <i>list</i></td><td>A list of design elements, input ports/pins, output ports/pins, and nets to which this strategy is applied.</td></tr> <tr> <td><b>-clamp_value</b> &lt;0   1   latch   Z&gt;</td><td>The value to which the input or output shall be clamped. The default is 0.</td></tr> <tr> <td><b>-applies_to</b> &lt;inputs   outputs   both&gt;</td><td>Whether the domain's input ports, output ports, or both are isolated. The default is <b>outputs</b>.</td></tr> </table> | <i>isolation_name</i> | Isolation strategy name. | <b>-domain</b> <i>domain_name</i> | The domain for which this strategy is applied. | <b>&lt;-isolation_power_net net_name  <br/> -isolation_ground_net net_name  <br/> -isolation_power_net net_name<br/> -isolation_ground_net net_name  <br/> -no_isolation&gt;</b> | The supply net(s) used to supply the isolation logic inferred by this strategy. Does not isolate the port, pin, or design element specified in the elements list. | <b>-elements</b> <i>list</i> | A list of design elements, input ports/pins, output ports/pins, and nets to which this strategy is applied. | <b>-clamp_value</b> <0   1   latch   Z> | The value to which the input or output shall be clamped. The default is 0. | <b>-applies_to</b> <inputs   outputs   both> | Whether the domain's input ports, output ports, or both are isolated. The default is <b>outputs</b> . |
| <i>isolation_name</i>                                                                                                                                                            | Isolation strategy name.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |                       |                          |                                   |                                                |                                                                                                                                                                                  |                                                                                                                                                                   |                              |                                                                                                             |                                         |                                                                            |                                              |                                                                                                       |
| <b>-domain</b> <i>domain_name</i>                                                                                                                                                | The domain for which this strategy is applied.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |                       |                          |                                   |                                                |                                                                                                                                                                                  |                                                                                                                                                                   |                              |                                                                                                             |                                         |                                                                            |                                              |                                                                                                       |
| <b>&lt;-isolation_power_net net_name  <br/> -isolation_ground_net net_name  <br/> -isolation_power_net net_name<br/> -isolation_ground_net net_name  <br/> -no_isolation&gt;</b> | The supply net(s) used to supply the isolation logic inferred by this strategy. Does not isolate the port, pin, or design element specified in the elements list.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |                       |                          |                                   |                                                |                                                                                                                                                                                  |                                                                                                                                                                   |                              |                                                                                                             |                                         |                                                                            |                                              |                                                                                                       |
| <b>-elements</b> <i>list</i>                                                                                                                                                     | A list of design elements, input ports/pins, output ports/pins, and nets to which this strategy is applied.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |                       |                          |                                   |                                                |                                                                                                                                                                                  |                                                                                                                                                                   |                              |                                                                                                             |                                         |                                                                            |                                              |                                                                                                       |
| <b>-clamp_value</b> <0   1   latch   Z>                                                                                                                                          | The value to which the input or output shall be clamped. The default is 0.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |                       |                          |                                   |                                                |                                                                                                                                                                                  |                                                                                                                                                                   |                              |                                                                                                             |                                         |                                                                            |                                              |                                                                                                       |
| <b>-applies_to</b> <inputs   outputs   both>                                                                                                                                     | Whether the domain's input ports, output ports, or both are isolated. The default is <b>outputs</b> .                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |                       |                          |                                   |                                                |                                                                                                                                                                                  |                                                                                                                                                                   |                              |                                                                                                             |                                         |                                                                            |                                              |                                                                                                       |
| <b>Return value</b>                                                                                                                                                              | Return 1 if it succeeds and 0 if it fails.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |                       |                          |                                   |                                                |                                                                                                                                                                                  |                                                                                                                                                                   |                              |                                                                                                             |                                         |                                                                            |                                              |                                                                                                       |

The **set\_isolation** command specifies the elements in the domain to isolate using the specified strategy — the isolation enable signal common to all, clamp value, and location.

**-isolation\_supply\_nets** can specify a single power net, a single ground net, or both. If only an isolation power net is specified, then the primary ground serves as the

isolation ground. If only an isolation ground net is specified, then the primary power net serves as the isolation power.

At least one of **-isolation\_power\_net** or **-isolation\_ground\_net** shall be specified, unless **-no\_isolation** is specified. If only **-isolation\_power\_net** is specified, the primary ground net shall be used as the isolation ground supply. If only **-isolation\_ground\_net** is specified, the primary power net shall be used as the isolation power supply. If both are specified, then these options specify the supply nets to use as the isolation power and ground nets.

The isolation power and ground nets are automatically connected to the implicit isolation processes and, when **-clamp\_value** is *latch*, to the isolation value register described in section 5 of the full specification or to the power and ground pins of an isolation cell if an isolation cell is explicitly mapped with the **map\_isolation\_cell** command. The switched off and on semantics for the implicit isolation process and isolation register, if present, are as described in section 5 of the full specification.

If there are multiple isolation strategies for one domain then, per strategy, **-elements** can be used to specify the elements to isolate. If **-elements** is specified, the elements shall be in the *domain\_name*. If **-elements** directly specifies a port by name (not implicitly, by specifying the port's instance or an ancestor of that instance), then the isolation strategy shall apply to that port regardless of whether that port's mode matches the one specified by the **-applies\_to** option. When **-elements** is not specified, this is equivalent to using the elements list that defines the power domain.

**-clamp\_value** can be:

- logic 0
- logic 1
- *latch* (the value of the non-isolated port when the isolation signal becomes active)
- logic Z

The following also apply.

- This command never applies to *inout* ports.
- If **-no\_isolation** is specified, then all the ports specified in **-elements**, explicitly or implicitly, and matching the **-applies\_to** (mode) shall not be isolated.
- It is an error if **-no\_isolation** is specified with any other arguments other than **-domain**, **-elements**, or **-applies\_to**.
- It is an error if more than one power or ground net is specified.
- It is an error if multiple isolation strategies specify the same design element, pins, ports, or nets.

Syntax example:

```
set_isolation outputs_only
-domain PD1
-isolation_power_net VDDbackup
-clamp_value 1
-applies_to outputs
```

## B.10 set\_isolation\_control

|                     |                                                                                                                                                                                                                           |
|---------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Purpose</b>      | Specify the control signals for a previously defined isolation strategy                                                                                                                                                   |
| <b>Syntax</b>       | <b>set_isolation_control</b> <i>isolation_name</i><br>-domain <i>domain_name</i><br>-isolation_signal <i>signal_name</i><br>[-isolation_sense <high   low>]<br>[-location <self   parent   sibling   fanout   automatic>] |
| <b>Arguments</b>    | <i>isolation_name</i> Isolation strategy name.                                                                                                                                                                            |
|                     | -domain <i>domain_name</i> The domain where the strategy applies.                                                                                                                                                         |
|                     | -isolation_signal <i>signal_name</i> The signal that causes the specified element to drive its clamp value.                                                                                                               |
|                     | -location <self   parent   sibling   fanout   automatic>    Where the isolation cells are placed in the logic hierarchy. The default is <b>automatic</b> .                                                                |
| <b>Return value</b> | Return 1 if it succeeds and 0 if it fails.                                                                                                                                                                                |

The **set\_isolation\_control** command allows the specification of the isolation control signal and sense separate from the **set\_isolation** command for those situations where the isolation strategy is known, but the control signals are not known until later.

Excepting the **set\_isolation\_control** command is executed within the current scope, and the addition of the **-location** option, the semantics here are equivalent to having specified the isolation control signal and sense with the **set\_isolation** command.

**-location** defines where the isolation cells are placed in the logic hierarchy.

- a) **self** – the isolation cell is placed inside the model/cell being isolated.
- b) **parent** – the isolation cell is placed in the parent of the cell /model being isolated.

- c) **sibling** – a new sibling is created into which the isolation cells are placed.
- d) **fanout** – isolation occurs at all fanout locations (sinks) of the port being isolated.
- e) **automatic** – the implementation tool is free to choose the appropriate locations (the default).

Syntax example:

```
set_isolation outputs_only
-domain PD1
-isolation_power_net VDDbackup
-clamp_value 1
-applies_to outputs

set_isolation_control outputs_only
-domain PD1
-isolation_signal cpu_iso
-isolation_sense low
-location parent
```

## B.11 set\_level\_shifter

|                     |                                                                                                                                                                                                                                                                                             |
|---------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Purpose</b>      | Specify a level shifter strategy                                                                                                                                                                                                                                                            |
| <b>Syntax</b>       | <pre>set_level_shifter level_shifter_name   -domain domain_name   [-elements list] [-applies_to &lt;inputs   outputs   both&gt;]   [-threshold value] [-rule &lt;low_to_high   high_to_low   both&gt;]   [-location &lt;self   parent   sibling   fanout   automatic&gt;] [-no_shift]</pre> |
| <b>Arguments</b>    | <i>level_shifter_name</i> Level shifter strategy name (used only for reporting).                                                                                                                                                                                                            |
|                     | <b>-domain</b> <i>domain_name</i> The domain for which this strategy is applied.                                                                                                                                                                                                            |
|                     | <b>-elements</b> <i>list</i> A list of design elements, pins, ports, or nets to which this strategy is applied.                                                                                                                                                                             |
|                     | <b>-applies_to</b> < <b>inputs</b>   <b>outputs</b>   <b>both</b> >      Whether the domain's input ports, output ports, or both are level shifted. The default is <b>both</b> .                                                                                                            |
|                     | <b>-threshold</b> <i>value</i> The voltage threshold (in volts) for determining when level shifters are required. The default is 0.                                                                                                                                                         |
|                     | <b>-rule</b> < <b>low_to_high</b>   <b>high_to_low</b>   <b>both</b> >      Which type of level shifters are required. The default is <b>both</b> .                                                                                                                                         |
|                     | <b>-location</b> < <b>self</b>   <b>parent</b>   <b>sibling</b>   <b>fanout</b>   <b>automatic</b> >      Where the level shifter is placed in the logic hierarchy. The default is <b>automatic</b> .                                                                                       |
|                     | <b>-no_shift</b> Can be specified with the <b>-elements</b> option to prevent the insertion of level shifters on the specified ports/pins and nets.                                                                                                                                         |
| <b>Return value</b> | Return 1 if it succeeds and 0 if it fails.                                                                                                                                                                                                                                                  |

The **set\_level\_shifter** command can be used to set a strategy for level shifting during implementation. *Level shifters* are placed on signals that have sources and sinks operating at different voltages, as their associated design elements are connected to different supply nets. If a level shifter strategy is not specified on a particular power domain, the default level shifter strategy consists of all elements in the power domain and uses the default strategy settings.

If **-elements** is specified, the elements shall be in the *domain\_name*. If **-elements** is used to specify a port or pin, a level shifter is inserted on that port regardless of any

**-threshold** or **-rule** specifications. The **-threshold** specification defines how large the voltage difference between the driver and sink needs to be before level shifters are inserted. Normally, this threshold is determined from the cell libraries; use this option to override the library values.

**-rule** can be **low\_to\_high**, **high\_to\_low**, or **both**. If **low\_to\_high** is specified, signals going from a lower voltage to a higher voltage get a level shifter when the voltage difference exceeds that specified by **-threshold**. If **high\_to\_low** is specified, signals going from a higher voltage to a lower voltage get a level shifter when the voltage difference exceeds that specified by **-threshold**. If **both** is specified, it is equivalent to having specified both rules in the strategy.

**-location** defines where the level shifter cells are placed in the logic hierarchy. All necessary supplies need to be available in the specified location.

- a) Self – the level shifter cell is placed inside the model/cell being shifted.
- b) Parent – the level shifter cell is placed in the parent of the cell /model being shifted.
- c) Sibling – a new sibling is created into which the level shifter cells are placed.
- d) Fanout – level shifter occur at all fanout locations (sinks) of the port being shifted.
- e) Automatic – the implementation tool is free to choose the appropriate locations.

The following also apply.

- This command never applies to *inout* ports.
- It is an error if the specified location is not within the logic design starting at the design root.

Syntax example:

```
set_level_shifter shift_up
  -domain PowerDomainZ
  -applies_to outputs
  -threshold 0.02
  -rule both
```

## B.12 set\_retention

|                     |                                                                                                                                                                                                                                                    |
|---------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Purpose</b>      | Specify which registers in the domain need to be retention registers and set the save and restore signals for the retention functionality                                                                                                          |
| <b>Syntax</b>       | <pre>set_retention retention_name -domain domain_name &lt;-retention_power_net net_name   -retention_ground_net net_name   -retention_power_net net_name -retention_ground_net net_name&gt; [-elements list]</pre>                                 |
| <b>Arguments</b>    | <div> <i>retention_name</i>      Retention strategy name (used only for reporting). </div>                                                                                                                                                         |
|                     | <div> <b>-domain</b> <i>domain_name</i>      The domain for which this strategy is applied. </div>                                                                                                                                                 |
|                     | <div> <b>&lt;-retention_power_net net_name   -retention_ground_net net_name   -retention_power_net net_name -retention_ground_net net_name&gt;</b>      The supply net(s) used to supply the retention registers inferred by this strategy. </div> |
| <b>Arguments</b>    | <div> <b>-elements list</b>      A list of objects in the power domain: design elements, named processes, or sequential <code>reg</code> or signal names to which this strategy is applied. </div>                                                 |
| <b>Return value</b> | Return 1 if it succeeds and 0 if it fails.                                                                                                                                                                                                         |

The `set_retention` command specifies which registers in the domain need to be retention registers and identifies the save and restore signals for the retention functionality. Only the registers implied in the elements list shall be provided retention capabilities. If a design element is specified, then all registers within the design element acquire the specified retention strategy. If a process is specified, then all registers inferred by the process acquire the specified retention strategy. If a `reg`, signal, or variable is specified and that object is a sequential element, then the implied register acquires the specified retention strategy. Any specified `reg`, signal, or variable that does not infer a sequential element shall not be changed by this command.

At least one of `-retention_power_net` or `-retention_ground_net` shall be specified. If only `-retention_power_net` is specified, the primary ground net shall be used as the retention ground supply. If only `-retention_ground_net` is specified, the primary power

net shall be used as the retention power supply. If both are specified, then these options specify the supply nets to use as the retention power and ground nets.

The retention power and ground nets are automatically connected to the implicit save and restore processes and shadow register or to the power and ground pins of a retention cell when that retention cell is explicitly mapped with the **map\_retention\_cell** command. The switched off and on semantics for the implicit save and restore processes and shadow register are as described in section 5 of the full specification.

If **-save\_signal** is specified, then **-restore\_signal** shall be specified. If **-save\_signal** is not specified, then **-restore\_signal** shall not be specified. If the save and restore signals are not specified, then they shall be specified in a **set\_retention\_control** command.

If **-elements** is specified, the elements shall be in the *domain\_name*. When **-elements** is not specified, this is equivalent to using the elements list that defines the power domain.

The following also apply.

- It is an error if *domain\_name* does not indicate a previously created power domain.
- It is an error if more than one power or ground net is specified.

Syntax example:

```
set_retention my_retention
-domain PDA
-retention_power_net volt_high
```

## B.13 set\_retention\_control

|                     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
|---------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Purpose</b>      | Specify the control signals and assertions for a previously defined retention strategy                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| <b>Syntax</b>       | <pre> <b>set_retention_control</b> <i>retention_name</i>   -<b>domain</b> <i>domain_name</i>   -<b>save_signal</b> {{<i>net_name</i> &lt;<b>high</b>   <b>low</b>   <b>posedge</b>   <b>negedge</b>&gt;}}   -<b>restore_signal</b> {{<i>net_name</i> &lt;<b>high</b>   <b>low</b>   <b>posedge</b>   <b>negedge</b>&gt;}}   [-<b>assert_r_mutex</b> {{<i>net_name</i> &lt;<b>high</b>   <b>low</b>   <b>posedge</b>   <b>negedge</b>&gt;}}]*   [-<b>assert_s_mutex</b> {{<i>net_name</i> &lt;<b>high</b>   <b>low</b>   <b>posedge</b>   <b>negedge</b>&gt;}}]*   [-<b>assert_rs_mutex</b> {{<i>net_name</i> &lt;<b>high</b>   <b>low</b>   <b>posedge</b>   <b>negedge</b>&gt;}}]* </pre> |
| <b>Arguments</b>    | <i>retention_name</i> Retention strategy name (used only for reporting).                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
|                     | <b>-domain</b> <i>domain_name</i> The domain for which this strategy is applied.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
|                     | <b>-save_signal</b> <i>save_net</i> The signal that causes the register values to be saved into the shadow registers.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
|                     | <b>-restore_signal</b> <i>restore_net</i> The signal that causes the register values to be restored from the shadow registers.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
|                     | <b>-assert_r_mutex</b> The restore signal for assertion.<br>{{ <i>net_name</i> < <b>high</b>   <b>low</b>   <b>posedge</b>   <b>negedge</b> >}}                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
|                     | <b>-assert_s_mutex</b> The save signal for assertion.<br>{{ <i>net_name</i> < <b>high</b>   <b>low</b>   <b>posedge</b>   <b>negedge</b> >}}                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
|                     | <b>-assert_rs_mutex</b> Both signals (save and restore) for assertion.<br>{{ <i>net_name</i> < <b>high</b>   <b>low</b>   <b>posedge</b>   <b>negedge</b> >}}                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| <b>Return value</b> | Return 1 if it succeeds and 0 if it fails.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |

The **set\_retention\_control** command allows the specification of the retention control signal and sense separate from the **set\_retention** command for those situations where the retention strategy is known, but the control signals are not known until later. As the assertions are related to the save and restore signals, they can also be specified with this command.

Excepting the **set\_retention\_control** command is executed within the current scope, the semantics here are equivalent to having specified the retention control signals, senses, and assertions with the **set\_retention** command.

The **set\_retention** command can also be used to specify any assertion options. Each option creates one or more assertions, which verification tools can trigger when the indicated RTL signals are active simultaneously. If **-assert\_rs\_mutex** does not specify a list of signals, this indicates the save and restore signals themselves are mutually exclusive.

The following also apply.

- The save signal shall be an existing net, port, or pin in the design.
- The restore signal shall be an existing net, port, or pin in the design.

Syntax example:

```
set_retention my_retention_strategy
-domain PDA

set_retention_control my_retention_strategy
-domain PDA
-save_signal {power_controller_inst/save_1 high}
-restore_signal {power_controller_inst/restore_1 low}
-assert_rs_mutex {clock_a posedge}
```

## B.14 set\_scope

|                     |                                                                                                                                                                                                            |
|---------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Purpose</b>      | Specify the current UPF scope                                                                                                                                                                              |
| <b>Syntax</b>       | <b>set_scope</b> <i>instance</i>                                                                                                                                                                           |
| <b>Arguments</b>    | <i>instance</i> The instance that becomes the current scope upon completion of the command.                                                                                                                |
| <b>Return value</b> | Returns the current scope prior to execution of the command as a full path string relative to the current design top if successful and the null string if it fails (e.g., if the instance does not exist). |

If the **set\_scope** command is called with no arguments or the UPF scope is not set, the scope is set to the top-level design.

If *instance* is **.**, the scope remains at the current instance. If *instance* is **..**, the context is moved up one level in the instance hierarchy. If *instance* begins with **/**, the scope returns to the instance whose name follows the **/**, relative to the design top.

Syntax examples:

```
set_scope foo/bar
```

```
set_scope ..
```

# *Glossary*

**Power Domain:** A collection of design elements that share a single primary supply connection and, at least conceptually, share a common power strategy.

**Isolation:** Isolation is a technique for controlling the behavior of a signal that is driven by a powered down power domain. Isolation consists of driving the signal to a known state - 1, 0, or latching it to a previous value when the power domain is powered down.

**Retention:** Retention is a technique for retaining the state value of registers in a powered down power domain.

**Isolation Cells:** Cells (gates) that perform the isolation function in a design. Also known as clamp cells or clamps.

**Level Shifters:** Cells (typically buffers) that translate inputs with one voltage swing to an output with a different voltage swing.

**Retention Register:** A register that extends the functionality of a normal register (flip-flop) with the ability to retain its memory during power down, assuming an appropriate second (always on) supply as well as save and restore signaling.

**Shadow Register:** The section of a retention register retains the register state during power down. Also known as a balloon registers (due to the topology of some implementations).

**Power Switch:** At the RTL and architectural level, a power switch allows the power to a power domain to be switched on or off (also known as power gated).

**Switching Network:** Physically, the power switch is implemented as a switching network of transistors. Also known as a switching fabric.

**Switching Transistor:** The individual switching transistor that makes up the switching network. Also known as a sleep transistor.

# Low Power Methodology Manual

## For System-on-Chip Design

*"Tools alone aren't enough to reduce dynamic and leakage power in complex chip designs - a well-planned methodology is needed. Following in the footsteps of the successful Reuse Methodology Manual (RMM), authors from ARM and Synopsys have written this Low Power Methodology Manual (LPMM) to describe [such] [a] low-power methodology with a practical, step-by-step approach."*

**Richard Goering**, Software Editor, EE Times

*"Excellent compendium of low-power techniques and guidelines with balanced content spanning theory and practical implementation. The LPMM is a very welcome addition to the field of low power SoC implementation that has for many years operated in a largely ad-hoc fashion."*

**Sujeeth Joseph**, Chief Architect - Semiconductor & Systems Solutions Unit, Wipro Technologies

*"The LPMM enables broader adoption of aggressive power management techniques based on extensive experience and silicon example with real data that every SOC designer can use to meet the difficulties faced in managing the power issues in deep submicron designs."*

**Anil Mankar**, Sr. VP Worldwide Core Engineering and Chief Development Officer, Conexant Systems Inc.

*"Managing power, at 90nm and below, introduces significant challenges to design flow. The LPMM is a timely and immediately useful book that shows how combination of tools, IP and methodology can be used together to address power management."*

**Nick Salter**, Head of Chip Integration, CSR plc,

### ABOUT THE AUTHORS:

**Michael Keating** is a Synopsys Fellow in the company's Advanced Technology Group, focusing on IP development methodology, hardware and software design quality and low power design.

**David Flynn** is an ARM R&D Fellow and has been with the company since 1991, specializing in low power System-on-Chip IP deployment and methodology.

**Robert Aitken** is an ARM R&D Fellow. His areas of responsibility include memory architecture, design for testability and design for manufacturability.

**Alan Gibbons** is a Principal Engineer at Synopsys, with a focus on development of advanced methodology and technology for ARM processor-based system design.

**Kaijian Shi** is a Principal Consultant in the Professional Services Group of Synopsys, specializing in low power design methodology and implementation.

**ARM**  **SYNOPSYS**

Advancing Design Solutions



**Springer**

springer.com

ISBN 978-0-387-71818-7



9 780387 718187