Design for Power Gating – and what UPF can, and cannot, do for you!

David Flynn

Research and Development
ARM Ltd
Cambridge, UK

www.arm.com

and

Visiting Professor, Pervasive Systems Research Centre
Southampton University
Southampton SO17 1BJ, UK

www.ecs.soton.ac.uk

ABSTRACT

Power gating is a valuable technique for reducing standby power in portable applications and comprises power rail switching of subsystems to cut leakage power. Power gating at the hardware level is now well supported in Synopsys design and verification tools, and the new UPF aware design flows are able to build on these. However from a system level design perspective inferring power gating, interface isolation and optionally state retention is not enough. This paper describes power management approaches for power gating with due attention to clocks, resets, testability and safe power sequencing - based on silicon proven results.

Table of Contents

Table of Figures

1 Introduction

Leakage power dissipation grows every generation[1] of CMOS process technology.[1] This leakage power is not only a serious challenge to battery powered or portable products but increasingly an issue that has to be addressed in mains-powered or tethered equipment too where added leakage power generates increased heat and often requires specialized packaging or cooling.[2][3]
It is highly desirable to add mechanisms to fully or partially switch the leaky power rails to reduce the leakage power that is dissipated whenever a subsystem is not actively in use. The Unified Power Format specification[4], UPF[2], provides the ability to be able to address the basic inference of power gating on an RTL design. However this needs care because the "golden source" is no longer just the configured RTL but also overlaid by functionality added from the power intent. Both must be designed and verified together.

- Chapter 2 describes the basics of power gating to set the background for this paper in terms of UPF constructs to gate power and clamp signals at interfaces.
- Chapter 3 addresses the complications of practical power gating to ensure safe turn-on, as well as addressing testability
- Chapter 4 introduces the problems of managing resets, clocks and clock gating
- Chapter 5 describes the additional complications of state retention with power gating
- Chapter 6 brings all this together with a worked example
- Finally, Conclusions are drawn as to best-practice approaches and pitfalls to avoid.

The work arises out of collaborative work between ARM R&D and Synopsys and the ongoing "SALT – Synopsys/ARM Leakage Technology" program. Results and experience from this collaboration underpin the work and Chapter 6 in particular.

2 The basics of Power Gating

This chapter describes the "Multi-Threshold" CMOS switching technique supported by the new UPF-aware EDA tool-chains:

- Power Gates – Header and Footer switches
- Clamps for isolating interfaces at power gated boundaries
- Buffer networks to control power gate networks and clamp control networks
- Describing basic power intent in UPF
- Basic power control sequencing

[1] High-K Metal-Gate process technology does help alleviate leakage power at 32nm but this will the problem will grow again at 28nm and 22nm nodes.
[2] UPF here relates to the specification 1.0 in this paper. Enhancements that will go into a 2.0 specification are not considered here.

2.1. Power Gates

A series transistor is inserted into either the supply or ground rail[3] to produce a switched "virtual" power rail that is used to power the standard cell logic.[5] Such transistors are not ideal or perfect switches:

- They introduce some switch resistance proportional to current drawn through them. Therefore the need to be large devices to minimize IR drop on the "virtual" switched supply rail for "on" current.
- As large devices they also have inherent and potentially "off" current so to achieve the best "I_{ON}" to "I_{OFF}" ratio, High thresh voltage, HVt, transistors are typically used.
- Buffer trees to control power gate networks and clamp control networks need to be powered from the un-switched supply rails.
- Rather than one single big conceptual switch a number of parallel switch cells[4] are usually deployed either in rings at the periphery or distributed across a power gated region in some checker-board or rows/columns approach.
- Ideally the switched rails are all joined in parallel (by a physical grid for example) to allow as much current sharing between switches as possible.

The basic "header" switch cell is based on a PMOS transistor that switches the supply rail (**Figure 1**); this requires an active-low enable signal, shown as **N_PWR** in this case. When low the residual leakage of the switch is in series with the standard cell logic that is being power gated.

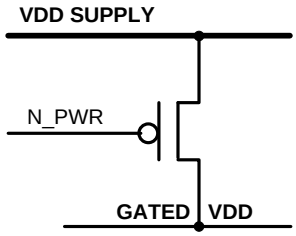

Figure 1: Header PMOS switch cell

Similarly the "footer" switch cell is based on an NMOS transistor that switches the supply rail (**Figure 2**). This requires an active-high enable signal, shown as **PWR** in this case.

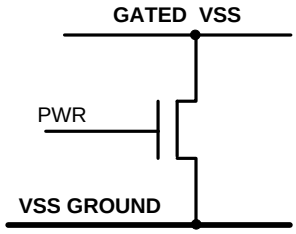

Figure 2: Footer NMOS switch cell

[3] Academic papers traditionally focus on adding switches to both power and ground rails symmetrically but the impact of two series switches compared to one in terms of the leakage power reduction when turned off mean that in normal commercial setting the cost of one switch is enough.

[4] In fact a power switch will typically be made up of a number of parallel switch "fingers" chosen for optimal I_{ON}/I_{OFF} ratio and current handling capacity.

A number of switch “strengths” may be provided in a library of power management components with a range of current-carrying capacities.

Header and Footer switches exhibit different ON and OFF efficiencies at different technology nodes so reusable designs need to abstract away the implementation choice.

2.1.1. Buffered Control Power Gates

As has been mentioned, the buffer tree that turns on and off the switched power network requires *un-switched* buffers to ensure these are correctly driven.
One approach to this is to include internal buffering within the power-gate switch structure. **Figure 3** shows this conceptually for header switch.

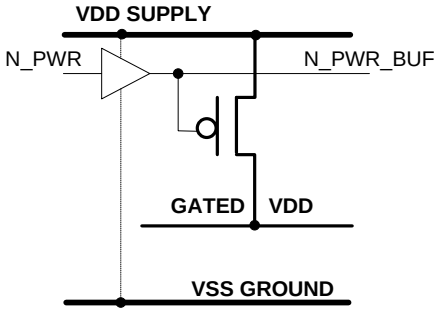

Figure 3: Buffered Header PMOS switch cell

Including buffering within the switch cell has an area cost but manages the power distribution inherently within the cell. Again a range of buffer strengths may be offered to support different switch topologies and interconnection lengths to avoid intermediate buffering.
The normal standard cell best-practice is to buffer both inputs and outputs in order to characterize the switching time internally independent of context-dependent output load. An example is shown in **Figure 4** and apparently appears attractive – but unlike the prior buffered design ends up with inverted control signaling so cannot be mixed and matched with raw unbuffered switches. (And doubly-inverting input and output is too area costly)

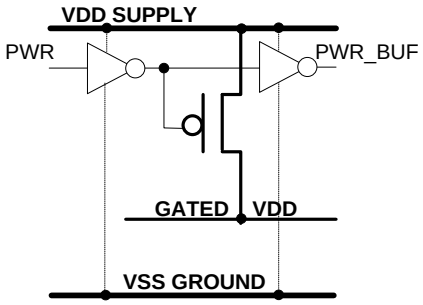

Figure 4: Alternative Buffered Header PMOS switch cell

Adding in buffering to power switch cells has the useful benefit of a characterized delay element in series with the power control element which can be used to control turn on switching as described in the next chapter.

2.2. Clamp Gates

Power gating a block of standard-cell logic usefully can reduce the standby leakage power significantly but any outputs from the block will present floating/non-logic level signals to powered regions that interface to them. Therefore interfaces need to be isolated logically to ensure crow-bar currents do not flow in down-stream transistor structures, potentially negating all leakage power savings.

- Power-gated Outputs are the primary concern. Such an output will typically float to some intermediate voltage with time – where the leakage off-currents through the power gates equal the leakage current of the power-gated cells. This will vary with temperature and process corner.
- Clamping such outputs to defined logic levels is a requirement for power gating. Conceptually the output needs to be logically forced to a valid logic level at the boundary. If an output has a wide logic fan-out then this is an obvious choice. In theory as all receiving inputs honour the clamping of power gated signals then these can be distributed – but any buffering that gets introduced must also clamp signals so may result in multiple series clamps.
- Isolation of inputs to a power-gated region is much less of a concern. Toggling inputs will waste some power but typically this will be limited to the first input stage of input port fan-in.

The simplest style of clamp is a basic transistor switch "shunt" that is turned on for an output whenever the block is power-gated. For "header"-switched power rails where the virtual rail collapses towards ground the natural choice is a pull-down (**Figure 5**) and for footer-switched virtual-ground implementations a pull-up is in order (**Figure 6**).

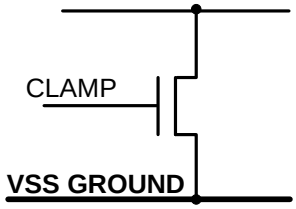

Figure 5: "Pull-down" conceptual clamp for Header-switched logic

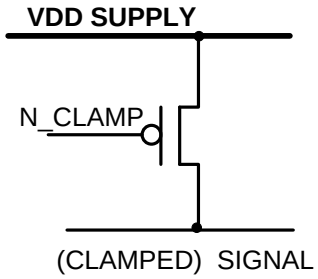

Figure 6: "Pull-up" conceptual clamp for Footer-switched logic

The logic value that signals are clamped to impacts on the interface protocols so the clamp values are not just an electrical but potentially a functional requirement.

In order to encourage design reuse and process portability the strong preference is for “logical” clamps that isolate interfaces with specific values regardless of whether header or footer implementation approaches are used at a particular process node.
A logical “AND” or “OR” function clamp is preferred – or in some specialist cases a “last-value” repeater style clamp can even be used – but with test and validation complications.
The “AND” style clamp is shown in **Figure 7** and an “OR” style clamp in **Figure 8**. In both instances the supply rail needs to be the un-switched power rails and the floating gate input is guaranteed by-design not to cause high internal current draw.

In these simple example clamp cells different logic polarity clamp-control signals have been introduced. Special cell variants may also be included in a power management control library which have both inverting and non-inverting control input flavours – but providing the library cell attributes make the function clear then clamp buffering can conceptually hide this from the designer by using inverting/non-inverting buffer trees. The buffering of the control signal, and the signal generation itself must of course be powered from the un-switched supply.

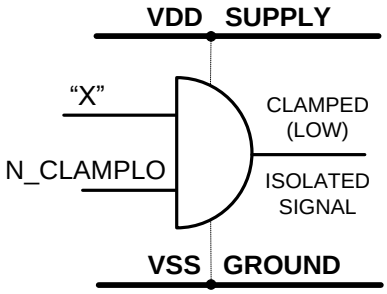

Figure 7: “CLAMP-LOW” isolation cell (with active-low CLAMP)

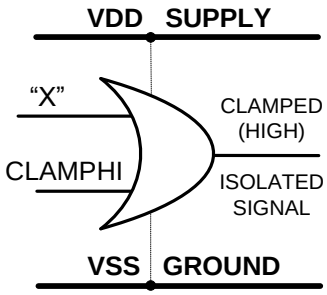

Figure 8: “CLAMP-HIGH” isolation cell (with active-high CLAMP)

So, clamps are necessary but do introduce a down-side in high-speed designs. A series gate, that is likely to be physically constrained by the voltage region boundaries will tend to have an impact on a near-critical path. The interface between a power-gated CPU and non-power-gated cache is an extreme example where a number of critical path signals do have to be clamped at the primary interface, requiring careful floor-planning and interface optimization.

Before UPF was standardized the designer typically had to instantiate isolation clamp cells explicitly at power gated interface boundaries. As a result a single port would often be forced on the implementation. Taking the cache boundary example again the only way of meeting timing would typically involving “punching through” multiple instances of the single RTL conceptual RAM clock to the cache memories.

The ability with UPF-style approaches to inferring isolation clamps allows the tools to do the optimizations based on physical parameters without having to manually "tune" the RTL by introducing explicit multiple clock ports and clamps, for example.

2.3. Power gating: switches and clamps together

Putting this all together the basic power gating functionality for an RTL subsystem can be summarized as:

- Defining the switched voltage region(s)
- Defining power control sequencing signal(s) for controlling the power-gating switch network
- Defining power control sequencing signal(s) for controlling (output) interface clamping.
- Defining the inactive protocol state for outputs – or system design requirement ensuring that all inputs on the fan-out from a clamped also factor in the CLAMP signalling to indicate that the interface is inhibited.
- Given a power control sequencer then the RTL subsystem can have the power gating leakage management inferred with a UPF side file.

2.3.1. Example UPF Power gating functionality inference

For a domain simply called "**A**" in this example the global and local power supplies are defined, a switched power rail (Header switches to the VDD supply rail in this case) and an **N_PWR** control signal is used as the power gating control signal. An example of mapping to specific header power gate cells is also shown at the end of this UPF code fragment:

```
create_power_domain TOP
create_power_domain A   -elements { uBlock }

create_supply_port  VDD -domain TOP
create_supply_net   VDD -domain TOP
connect_supply_net  VDD -ports VDD
create_supply_port  VSS -domain TOP
create_supply_net   VSS -domain TOP
create_supply_net   VSS -domain A -reuse
connect_supply_net  VSS -ports VSS
create_supply_port  VDDA -domain A
create_supply_net   VDDA -domain A
connect_supply_net  VDDA -ports VDDA

create_supply_net   VDDA_SW -domain A -resolve parallel

set_domain_supply_net TOP -primary_power_net VDD -primary_ground_net VSS
set_domain_supply_net A -primary_power_net VDDA_SW -primary_ground_net VSS

create_power_switch uswitch_PWR_A -domain A \
    -input_supply_port {VDDA VDDA } \
    -output_supply_port {VDDA_SW VDDA_SW } \
    -control_port {N_PWR N_PWR } \
    -on_state {on_state VDDA {!N_PWR }}

map_power_switch uswitch_PWR_A -domain A -lib_cell HEADBUF16_X1M_A12TL
```

2.3.2. Example UPF Clamp functionality inference

Inferring output clamps for domain "**A**" is shown where in the example the control signal is an active low signal called **N_CLAMP**:

```
set_isolation vsoc -domain A -isolation_power_net VDD \
```

```
    -isolation_ground_net VSS \
    -clamp_value 0 -applies_to outputs
set_isolation_control vsoc -domain A \
    -isolation_signal N_CLAMP  -isolation_sense low -location parent
```

2.3.3. UPF-inferred power gating circuitry

The resulting circuitry will be of the form shown conceptually in **Figure 9**. The power-gated region is denoted by voltage region "**A**" in the UPF description, the power switch control by **"N_PWR**" and the output clamping by "**N_CLAMP**".

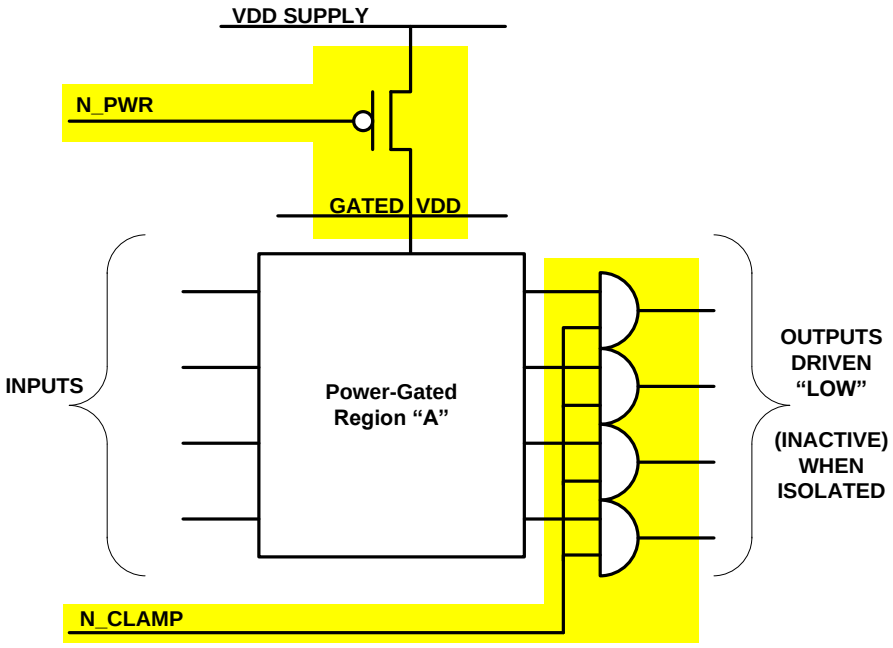

Figure 9: Example Power-Gated sub-system after UPF power intent inference

2.3.4. Example UPF Power State Table

The final requirement is to specify at the system level the valid power management states and these are described in the UPF as a very basic Power State Table. The code fragment below shows the simplest example for the example described so far with only a "**full-on**" and a "**dormant**" state:

```
add_port_state VSS -state {on 0.00}
add_port_state VDD -state {on 0.99}
add_port_state VDDA -state {on 0.99}
add_port_state uswitch_PWR_A/VDDA_SW -state {on 0.99} -state {off off}

create_pst    pstA           -supplies {VSS VDD VDDA  VDDA_SW}
add_pst_state fullon  -pst pstA -state { on  on   on       on}
add_pst_state dormant -pst pstA -state { on  on   on      off}
```

2.4. Example Power gating control sequencing

There are some basic requirements around controlling this inferred power gating. To power down:

- Shutting down the sub-system to be power-gated in an orderly manner

- e.g. stopping the clock(s) once current transactions are completed
- Clamping the outputs to quiescent, inactive, states
- Turning off the power

And to power back on again:

- Turning back on the power gates
- Resetting any internal clocked state
- Disabling the output clamps to expose initial restart state
- Re-enabling the clock(s)

Figure 10 shows example waveforms to implement the power control. In addition there must be some form of "wake-up" stimulus where timed or from some externally triggered event to know when to re-power and restart the subsystem.

For a microprocessor the embedded software has to cooperatively consent to going into power-down mode and is responsible for ensure that suitable external wake-up interrupt sources are left enabled – and not themselves power-gated!

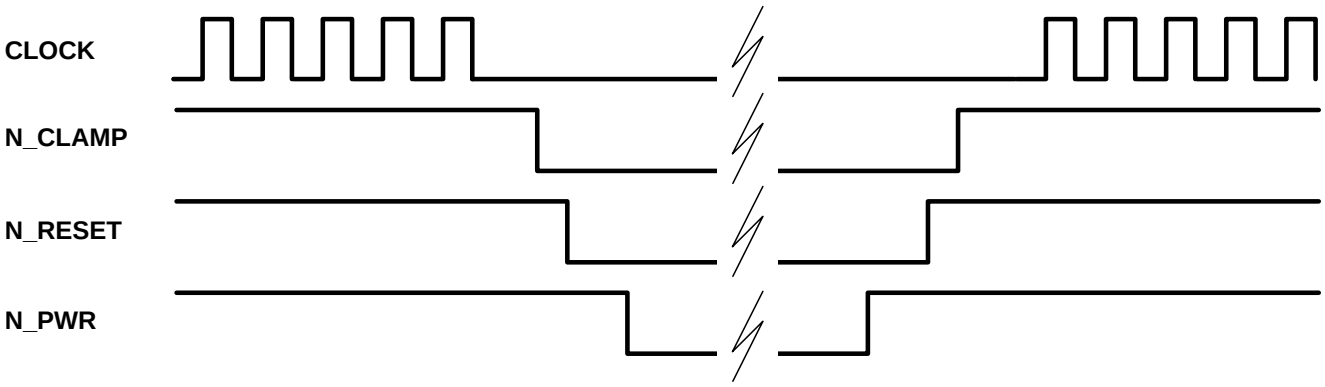

Figure 10: Basic power gating control sequence waveforms

In the case where synchronous rather than asynchronous reset functionality is all that is provided for registers in the cell library registers then a more complex sequence that enables a clock pulse while reset is asserted would be a requirement.

3 Power Gating in practice and production

Taking the theory and building block from the previous chapter, this chapter describes real-world complications of turning power gated systems on and off taking into account the physical switching current impact, as well as the issues of how to address testability of the power gating control networks.

The initial questions that arise are of the form:

- System response latency: How fast can a power-gated system be turned back on?
- Power supply integrity: Power-gating rush currents and effect on SOC ground/power rails?

- Testability: How to test for missing switches (e.g. broken control chains) or stuck-on power gates that would have a serious impact on product “standby” battery life?

More complex questions such as that of power rail decoupling are not covered in this paper as these are implementation challenges beyond the scope of UPF.

3.1. Managing the turn-on impact of Power-Gating

Minimizing the latency of switching on of power gated subsystems is a worthy aim but the following need to be considered:

- How large a region is to be power-gated as an entity? The larger the region the bigger the fan-out of power control buffering – and the greater the delay to turn it on.
- When power gated (off) a subsystem basically discharges the bulk capacitance of the switched rail cells themselves, plus any decoupling capacitors that have been inserted to improve the local power grid high-frequency response.

To minimize power gating turn-on delay a buffer tree approach looks desirable in order to get the shortest turn-on time. **Figure 11** shows this in simple conceptual terms. Now clock trees are typically optimized for minimal skew between final clocks. Rather than have every power gate turned on simultaneously the preference would be for a deliberately unbalanced tree with wider turn-on timing spread.

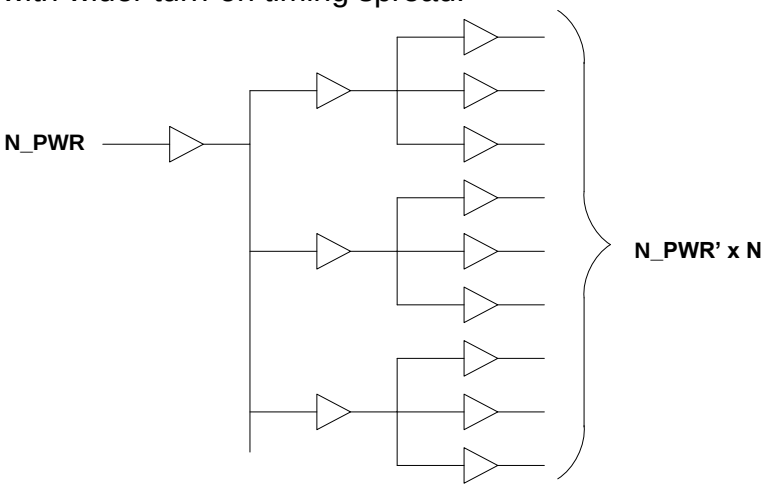

Figure 11: Conceptual Tree-buffered Power Control network

There are well known techniques for controlling power up to avoid major transients [6] and the concept of using a “weak” power switch network to start to bring up the voltage rail gently followed by enabling of the “main” switch network that provides the low-IR drop supply is easily supported in UPF, and with the turn-on current analysis performed with a tool such as PrimeRail.

In essence:

- Firstly turn on START power switch network that resistively “charges” the switched rail up to operating voltage in a delayed manner. This should be as slow as needed to avoid rail voltage drop that could cause loss of data in logic sharing the same unswitched power rail.

- Secondly, turn on the MAIN power network to provide the low resistance power gated connection to the power rail.
- The speed of the MAIN turn-on is not an issue – and ideally should be made fast to improve wake-up latency. (A tree-buffered network for this would not be a problem)

3.1.1. Example UPF Multi Switch Network Inference

A second switch array is added in parallel with the primary power switch array and in "weaker" power gate cells are mapped onto this network in this example, shown with an **N_START** control signal in addition to the primary switches controlled by the **N_PWR** control:

```
create_power_switch uswitch_START_A -domain A \
    -input_supply_port {VDDA VDDA } \
    -output_supply_port {VDDA_SW VDDA_SW } \
    -control_port {N_START N_START } \
    -on_state {on_state VDDA {!N_START }}

map_power_switch uswitch_START_A -domain A -lib_cell HEADBUF4_X1M_A12TL

create_power_switch uswitch_PWR_A -domain A \
    -input_supply_port {VDDA VDDA } \
    -output_supply_port {VDDA_SW VDDA_SW } \
    -control_port {N_PWR N_PWR } \
    -on_state {on_state VDDA {!N_PWR }}

map_power_switch uswitch_PWR_A -domain A -lib_cell HEADBUF16_X1M_A12TL
```

3.2. Addressing Power Gating network testability

Design-for-test guidelines are well understood for best-practice RTL design [8]. The primary requirements are:

- Externally controllable primary Clock(s)
- Externally controllable (override of) Clock Gating
- Externally controllable primary Reset(s)

However when power gating is inferred a number of extra requirements must be addressed in RTL control design:

- Power Gating controls (typically state machine register outputs) must be made controllable – and must be on for standard logic test functionality
- Power gating clamp controls (typically state machine register outputs) must be made controllable – and must be "pass-thru" for standard logic test functionality

Additionally it becomes desirable to provide a level of testability of the integrity of power gating control buffer networks:

- Visibility of start and end of control buffer chains for combinatorial logic test.

3.2.1. Example Verilog RTL coding for Testability

At a minimum, the power gating control signals must be forced to prevent them accidentally being asserted. In the example coding below the Power Control state machine outputs are made controllable during manufacturing test configuration (**test_mode** asserted in this case):

```
assign N_PWR_REQ     = (test_mode) ? 1'b1 : PwrState[1]; // force ON
assign N_ISOLATE_REQ = (test_mode) ? 1'b1 : PwrState[2]; // force PASS-THRU
assign N_RESET_REQ   = (test_mode) ? 1'b1 : PwrState[3]; // de-asserted
```

Alternatively making these explicitly controllable from external pins, even if these are shared between multiple power gated regions, then full test coverage patterns can be developed properly. If in **test_mode** a couple of input pads can be re-assigned as power gating control (**TEST_NPOWER**) and as clamping control (**TEST_NCLAMP**) in addition to the familiar controllable reset (**TEST_NRESET**) then example coding would be of the form:

```
assign N_PWR_REQ     = (test_mode) ? TEST_NPOWER : PwrState[1]; // controllable
assign N_ISOLATE_REQ = (test_mode) ? TEST_NCLAMP : PwrState[2]; // controllable
assign N_RESET_REQ   = (test_mode) ? TEST_NRESET : PwrState[3]; // controllable
```

Additional STIL test code can be used to insert test sequences to emulate entry and exit of power down states.

3.3. Request/Acknowledge handshakes for Power Gating control

In order to build designs that are portable and reusable it is ideal not to encode into the RTL fixed delays in terms of precise numbers of cycles – which could well need changing/validating every product generation depending on switch cell technology.

The approach adopted here is to work with the convention that the power network control buffering is treated as one or a number of daisy-chained (buffered) switch elements.

- Logical integrity of buffer chains can be made testable explicitly
- The turn-on behavior is inherently “phased” such that the rush currents of simultaneous switch turn-on are mitigated.
- The timing of long chains can be abstracted away in the implementation – although the “acknowledge” handshake forms the end of a power gating buffer chain must then be treated asynchronously (and typically synchronized in RTL)

Figure 12 depicts the desired functionality schematically, and there typically will be one or more. The interface signals are shown indexed here for where there is more than one control network – but indices are not supported in UPF so in fact these will require explicitly named bit-wide port names.

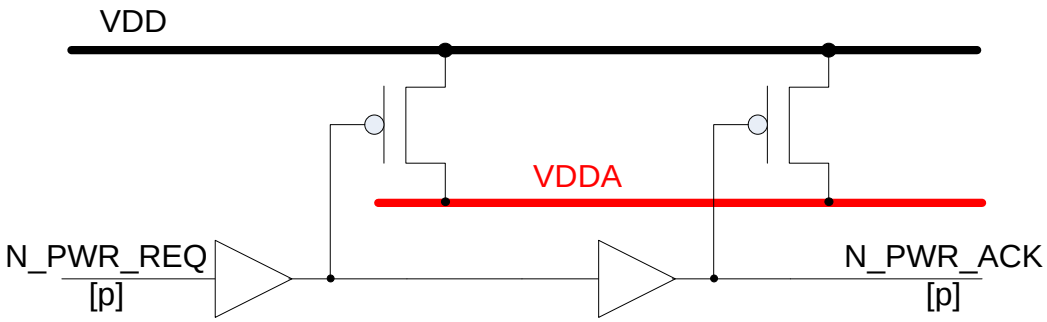

Figure 12: Power Gating control chain Request/Acknowledge

The behavior, shown as a timing waveform is shown in **Figure 13**.

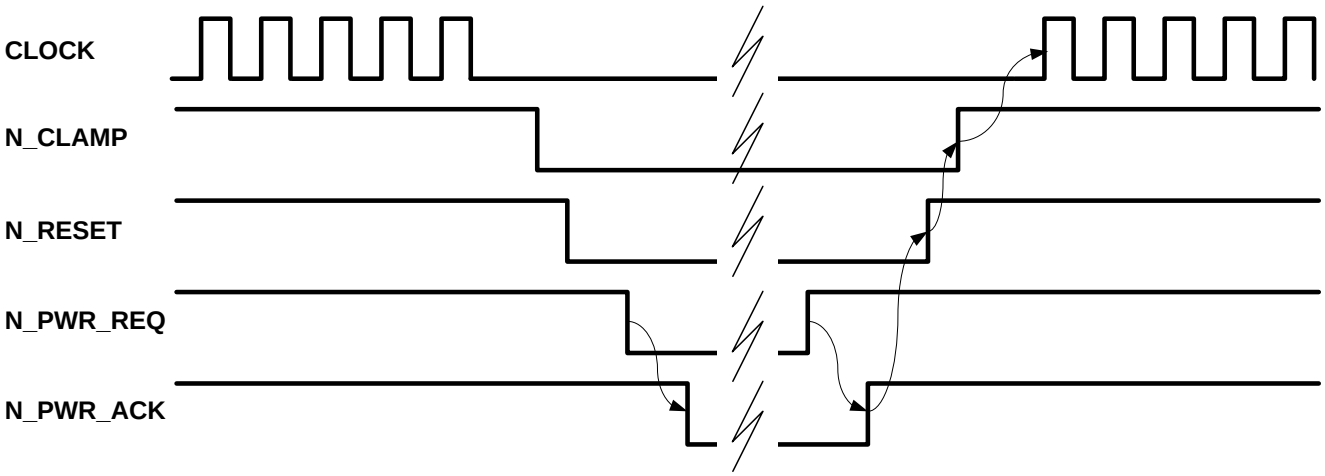

Figure 13: Power Gating Request/Acknowledge handshake usage

This double-ended handshake signaling then gives a clean system level interface to the power gated subsystem:

- When POWER_REQ and (synchronized) POWER_ACK are both valid, power is stable and logic usable
- After exit from power-gating the safe state machine "wake-up" sequencing is ideally depended on (synchronized) POWER_ACK being valid and POWER_REQ being valid. (The second term avoids a race condition when trying to re-awake before fully into powered down mode – which is unsafe as logic is only powered by the virtual rail decoupling capacitance)

3.3.1. Example UPF Power Request/Acknowledge functionality inference

UPF provides support for acknowledge signaling directly to implement such handshakes:

```
create_power_switch uswitch_PWR_A -domain A \
    -input_supply_port {VDDA VDDA } \
    -output_supply_port {VDDA_SW VDDA_SW } \
    -control_port {N_PWR N_PWR } \
    -ack_port {N_PWR_ACK N_PWR_ACK {N_PWR_REQ}} \
```

```
    -on_state {on_state A {!N_PWR }}
```

3.3.2. Example UPF Multi-chain Power Req/Ack functionality inference

Extending the principle to multiple power control chains to address the controlled start-up sequencing of power gating is straightforward:

```
create_power_switch uswitch_START_A -domain A \
    -input_supply_port {VDDA VDDA } \
    -output_supply_port {VDDA_SW VDDA_SW } \
    -control_port {N_START N_START } \
    -ack_port {N_START_ACK N_START_ACK {N_START_REQ}} \
    -on_state {on_state A {!N_START }}

create_power_switch uswitch_PWR_A -domain A \
    -input_supply_port {VDDA VDDA } \
    -output_supply_port {VDDA_SW VDDA_SW } \
    -control_port {N_PWR N_PWR } \
    -ack_port {N_PWR_ACK N_PWR_ACK {N_PWR_REQ}} \
    -on_state {on_state A {!N_PWR }}
```

4 Power Gating and Resets, Clocks and Clock Gating

This chapter introduces the problems of managing resets, clocks and clock gating that cannot be handled by UPF side-files. These must be addressed by the RTL designer.

The questions that need addressing here are all about:

- State loss and re-initialization after power gating?
- When and how can the clock be safely re-enabled after power gating?

4.1. Power Gating and Resets

Taking the conceptual control signal sequencing described so far, it would appear desirable and simple to reuse the Clamp control as the state reset mechanism for any power-gated register state that will have been lost/corrupted during power-down. (State Retention will be covered in the next chapter).

Figure 14 depicts this logical simplification. The active-low **N_CLAMP** signal that forces outputs to known states in this example is also used to force the active-low asynchronous **N_RESET** input whenever the power-gated sub-system is clamped ready for power-gating:

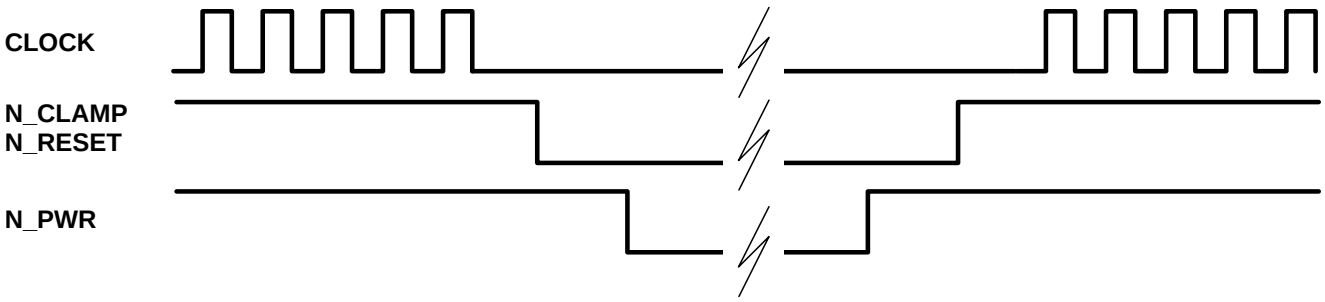

Figure 14: Conceptually combining output Clamp with RESET input control

The schematic view of what is desired is shown in **Figure 15**. As well as clamping the outputs (low) the local reset network is asserted active such that all state will be re-initialized correctly after waking-up from power-gating.

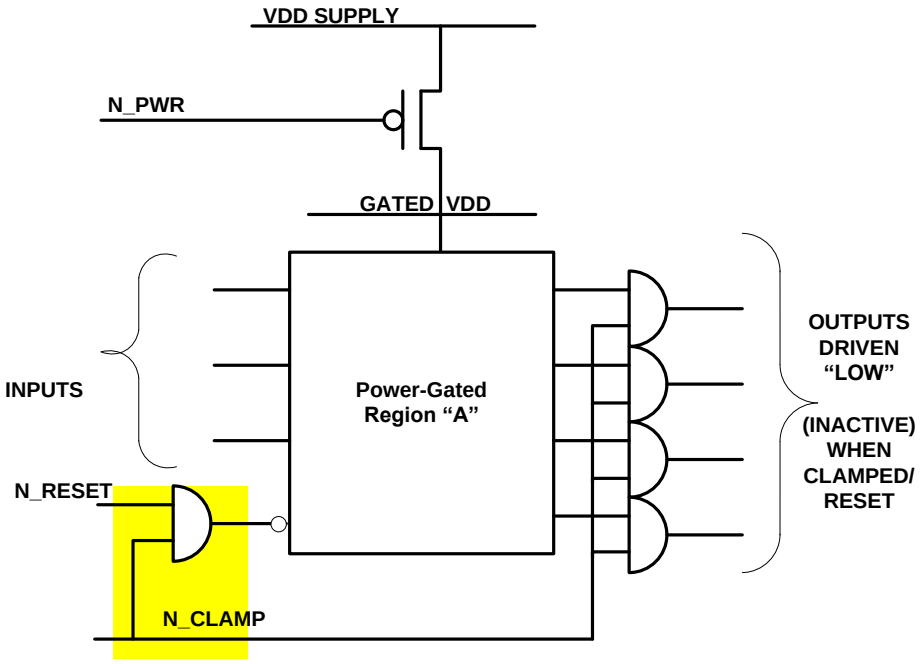

Figure 15: Schematic showing output Clamp with RESET input control

4.1.1. Issues with RESET clamping

Inferring the state re-initialization with UPF clamp structures requires care because the UPF is now starting to impact on the RTL behaviour as written and understood by the designer before power-intent was inferred.

The buffer tree latency balancing for the reset (input) tree and the output clamp buffering may potentially lead to surprises in static timing analysis. If the reset tree has a significant buffer depth and recovery-arc latency then the output timing becomes dependent on both the output clamp "enable" arc and the internal reset path dependent on the input "de-assert" clamp. Separating input and output clamp ports allows independent latency balancing.

In summary:

- Ideally maintain independent UPF-inferred control inputs for Input (reset) and Output clamping
- Handle carefully the case where the reset buffer tree latency is significant as this makes balancing timing closure on outputs a challenge.

4.2. Power Gating and Clocks/Clock-Gates

Extending the optimization of the conceptual control signal sequencing even further, in a system where there are primary RTL clocks that are not independently controllable for power-gated subsystems then it appears attractive to re-use the interface isolation signal to also inhibit clocks as well as re-initializing register state as well as the mandatory clamping of outputs.

Figure 16 shows this further logical simplification. The active-low **N_CLAMP** signal that forces outputs to known states in this example is not only used to force the active-low asynchronous **N_RESET** input whenever the power-gated sub-system is clamped ready for power-gating but also to inhibit the clocks.
The primary concern is that clocks should be inhibited on exit from power-gating so ensure clean behaviour – especially when state-retention is addressed in the following chapter.

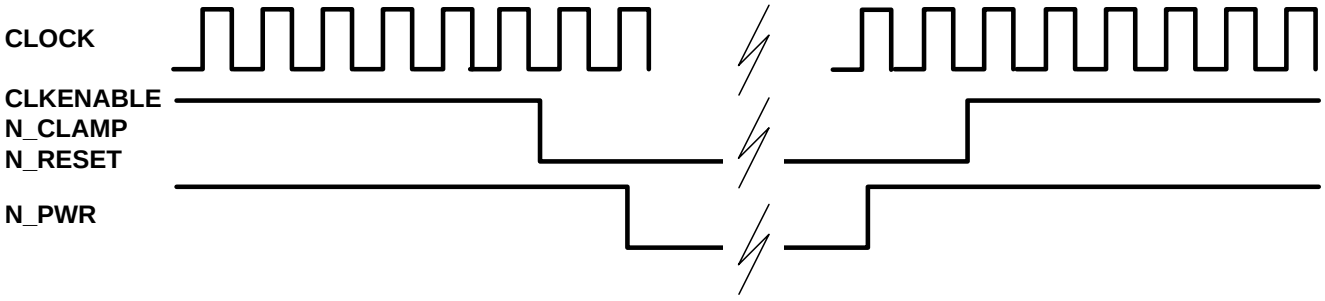

Figure 16: Conceptually combining output Clamp with RESET and CLOCK enable control

The schematic view of what is desired is shown in **Figure 17**. As well as clamping the outputs (low) the local clock is inhibited as well as the reset network being asserted active such that all state will be re-initialized correctly after waking-up from

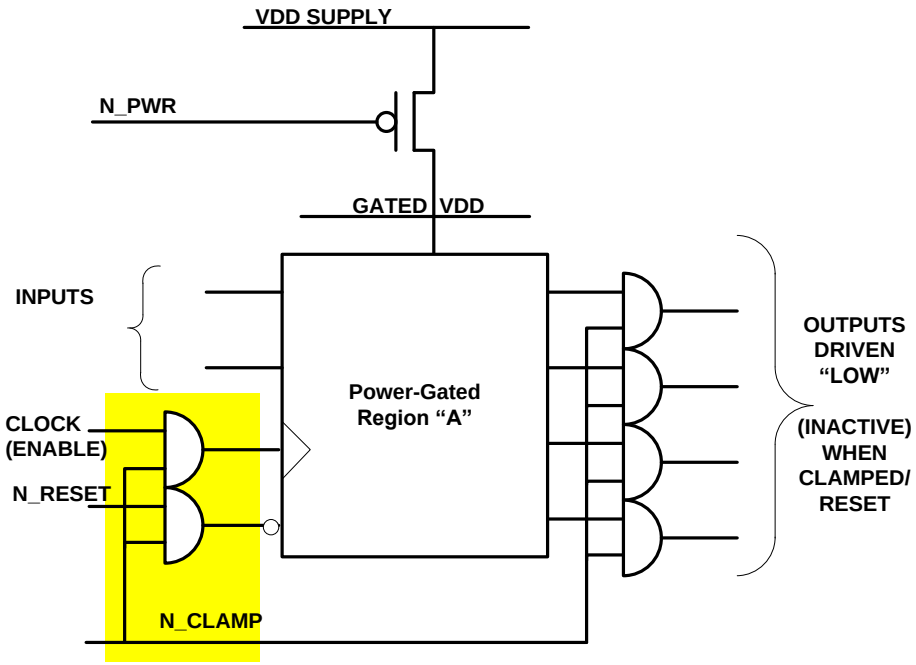

Figure 17: Schematic showing output Clamp with RESET and CLOCK input control

4.2.1. Issues with CLOCK clamping

Simply putting an AND-gate on a clock is not recommended best-practice. In order to avoid clock waveform truncation the "enable" signal must be timed to the appropriate phase of the clock – the low-phase in the examples used in this paper with industry-standard rising-edge clocks.

"Integrated Clock-Gating" (ICG) is conventionally the way this is handled cleanly within the implementation EDA flow. These cells provide an internal latch that samples during the phase of the clock that precedes the active-edge and internally gates the clock waveform safely. Such ICG components have test-override functionality as well to force clock enables during ATPG test sequences for example.
Exposing a top-level clock gate enable port is the cleanest way of supporting UPF-automated clock gating while the clamp signal is asserted.

Because clock buffer tree latency is typically significant there may in fact be real benefit in treating the clock enable (input) clamp and the output clamp as separate ports form a UPF perspective. This then allows independent latency balancing of the two controls even if they are conceptually driven by the same state machine control term.

In summary:

- Maintain independent UPF-inferred control inputs for clock enable and output clamping
- Provide explicit clock enable exposure to a subsystem that is to be power-gated to provide the cleanest mechanism for inhibiting local clock(s)

- Beware trying to use this technique on large subsystems with deep latency clock trees – especially if almost, or greater than, ½ clock cycle. Timing closure on input and output clamps cannot be managed from a single control signal.

5 Power Gating and State Retention

It is not the purpose of this paper to deal with the many intricacies of state retention with power gating, SRPG[9] suffice to say that the power gating control sequencing needs further attention – and any state retention inferred from UPF must be handled with care in the light of the RTL system design and verification.

UPF offers the designer the ability to add selective state retention independently through a side file. This is highly dangerous! It is quite possible to choose to retain arbitrary register state that could potentially deadlock the (sub)system or grow the verification space by orders of magnitude.

For legacy IP design re-use in a Power-Gating/UPF implementation the only safe options are:

- Total state retention. This effectively models the RTL designers intent where every registered state value is persistent between clock events, and the EDA tools that infer clock gating etc can make global optimizations based on all visible state. (e.g. all register state that is factored into a clock gating enable term is guaranteed valid)
- Zero-state retention. This maps onto the RTL design and reset/preset coding and sequencing that is verified from explicit initialization.

If both edges of the clock are used internally to the IP block – or the clock cannot be held cleanly inactive (low in the case of a rising edge clocked subsystem) on entry to and exit from power gating then *retention Clock Gating cells* must be available in the power management support cell library. This is because clock gates can only be “transparent” and re-evaluate their clock enable terms (clock low for rising-edge registers) for one phase of the clock. Any clock gates on falling-edge registers in this case must retain their latched enable state in order to behave correctly on state restoration.

For new IP designs it is strongly recommended that:

- “State retention islands” are explicitly coded in RTL hierarchical modules
- There are no combinatorial inputs to clock gating terms from non-retained register state
- Independent resets are coded in the RTL for retained and non-retained state – even if these eventually share a common external reset control signal
- Only one edge of the clock should be used if inference of specialized retention clock gates is to be avoided.

5.1. State Retention Power Gating control sequencing

The basic control state machine needs to address the following requirements:

- Reset control must be handled independent of Clamping
- State Save and Restore protocol sequencing must be added carefully

- A special power-on-reset initialization sequence is required in the start up case where retention state is invalid and where all retention registers must be reset fully
- In the case of selective state retention – where only some of the register state has hardware retention support then the reset restore sequencing must be handled especially carefully

Figure 18 illustrates an example control sequence for a state retention region with independent "**SAVE**" and "**RESTORE**" control signals. Similary **Figure 19** illustrates the same functionality with an alternative style of retention register which has a single "sample and hold" control interface, here shown as an active-low "**N_RETAIN**" signal.

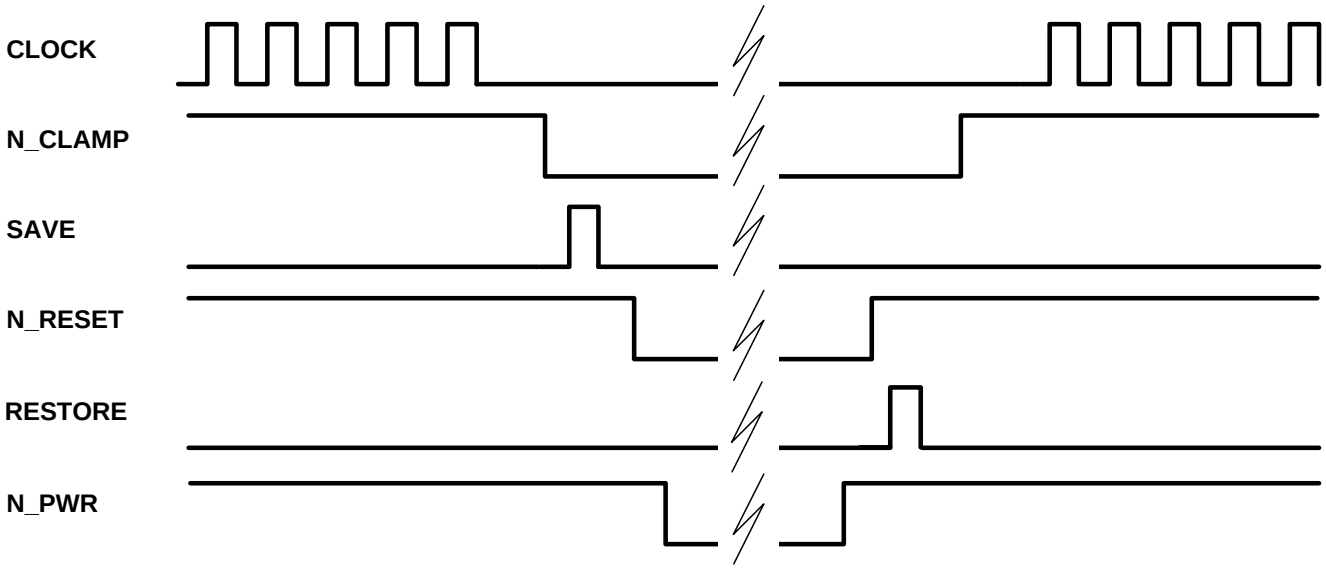

Figure 18: Power gating control with State Retention : separate SAVE/RESTORE controls

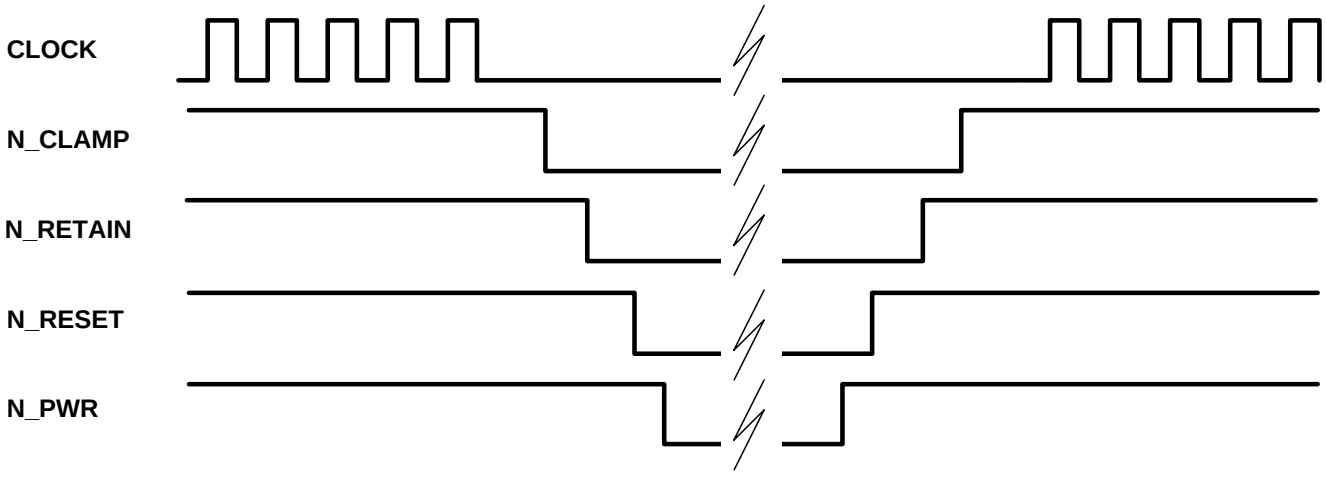

Figure 19: 'Power gating control with State Retention : single RETAIN control

5.1.1. Example UPF state retention inference

A coded example for state retention for all state is shown below. In this case the retention cells are mapped to a library subset by the name of "**DRFF**":

```
set_retention A_ret -domain A \
    -retention_power_net VDDA \
    -retention_ground_net VSS
set_retention_control A_ret -domain A \
    -save_signal {N_RETAIN low}
    -restore_signal {N_RETAIN high}
map_retention_cell A_ret -domain A \
    -lib_cell_type DRFF
```

In the less desirable case where clock gates with retention have to be inferred where by design they are not guaranteed to be "open" and re-evaluating the clock-gating terms from retained register state then specific coding is required. In this case an "**RLAT**" latch is being inferred, only for named elements:

```
map_retention_cell A_ret -domain A \
    -lib_cell_type RLAT \
    -elements {. . .}
```

5.1.2. Issues with UPF Inferred Retention

To summarize:

- Total state retention state matches the RTL environment used for design and verification of the RTL
- Zero state retention is easy
- Single edge clocking is strongly advised to avoid specialized Retention Clock Gating latch requirements
- Selective state retention requires complete re-verification of every system power state supported in the Power State Table.

Finally from the physical perspective, even more attention may have to be devoted to the safe power-on gating implementation. State Retention is only usable if each and every retained register is 100% guaranteed not to be corrupted. Any state bit corruption is potentially catastrophic and may cause malfunction or deadlock.
In systems that seek to implement fast wake-up from power gating there is potentially the chance that the precious state that has been saved on entry and preserved during power gating may end-up being corrupted due to a significant turn-on rush current that causes ground or supply rail bounce. The retention latch structure is designed for low retention leakage current so the devices may be sensitive to power transients.

- Ensure power-gating turn-on is managed especially carefully in subsystems that implement retention (using analysis tools such as PrimeRail™)

6 Worked example: Power Gating applied to a CPU

This chapter brings all this together with a worked example developed for ARM reference system designs where CPU subsystems are power-gated, with and without retention.

- Power controller design
- Power-gating start-up rush current analysis and measurement
- Addressing testability
- Distributed power switch topologies

Brief illustrations from the work at 90nm, 65nm and 45nm are used in this chapter.

6.1. Addressing Power Gating control

The control state machines developed for power control basically implement the sequences developed and described in this paper.

In order to ensure the RTL coding is reusable and does not have hard-coded numbers of clock cycles per sequence step (very technology and CPU size dependent) every output control signal is treated as a request signal. For every output there is an equivalent "acknowledge" signal which basically indicates valid assertion. Synchronizers are provided on all inputs such that they can be effectively asynchronous to the state machine clock and requesting signal.

- For the power-gating control buffer networks the UPF-inferred acknowledge outputs are fed back to the acknowledge inputs (reflecting the series buffer chain delays).
- For other nearly instantaneous signals the outputs are simply wired up as request to acknowledge with what is typically a two clock cycle synchronizer path.
- The master clock may be chosen to be the same as, synchronous to or asynchronous to the power-gated subsystem. In some systems this is driven from a SOC bus-clock that is kept alive when the processor and high speed clock are asleep. In others this simply runs off a 48MHz reference clock such that PLL's can be stopped for deeper power saving states.

Figure 20 shows the basic functionality at block level. The input acknowledge signals that are simply wired back to the request output signals, typically, as they do not have implementation-dependent latency, are shown in parentheses – e.g. (nRETAIN_ACK) is wired to nRETAIN_REQ output. But the general principal holds for adding a synchronizer on ever input to the state machine such that that these can be treated as asynchronous in designs with deeper CLAMP or RESET buffer tree latency for example.

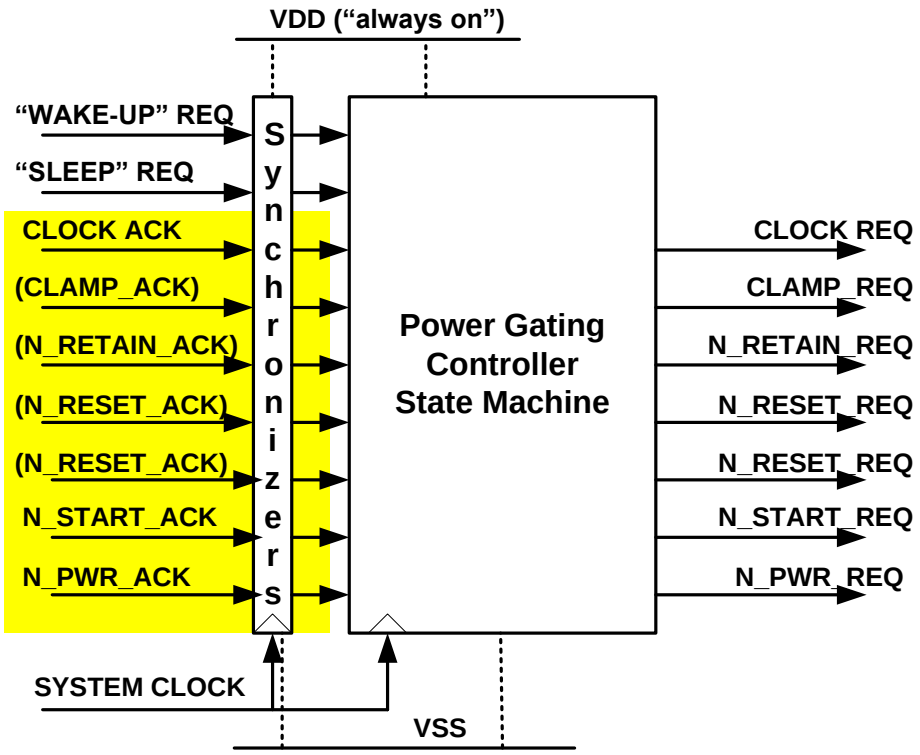

Figure 20: Power Gating State Machine with REQ/ACK handshake controls

In the case of the ARM CPU the “SLEEP” request is in the form of a Wait-For-Interrupt status signal that indicates the CPU has finished all internal processing as well as completed all external bus transactions[5].
The “WAKE” signal is system specific and typically is generated by the interrupt controller subsystem that is sensitized to level- or edge-triggered requests from peripherals that require service.

6.2. Addressing safe turn-on of Power Gating

A two-phase power gating approach has been used for reasonably complex cached CPU subsystems. In the case of the “SALT1” ARM926 project [7] implemented in a TSMC 90nm 1-volt “G” technology, a diagnostic mode was built into the test chip to allow external overriding of the “soft-start” control such that the main power gates are forced on directly.
Figure 21 shows the effect of forcing on the strong power gating directly while **Figure 22** shows the reduced current spike when the soft-start “rail pre-charge” is used. The waveforms show the current as measured across a low-impedance external series resistor in terms of a negative voltage spike that reflects the IR drop at the instant of power gating. Although the measurement approach is not sophisticated enough to take precise measurement values, the relative current surge peak is clearly demonstrable. (Recorded in real time using a digital storage oscilloscope).
Before tape-out some basic H-SPICE simulations of representative power-gated networks and decoupling capacitance were performed but analysis of the full design was not possible.

[5] The signal named STANDBYWFI on cached CPUs or ARM9 and ARM11 families

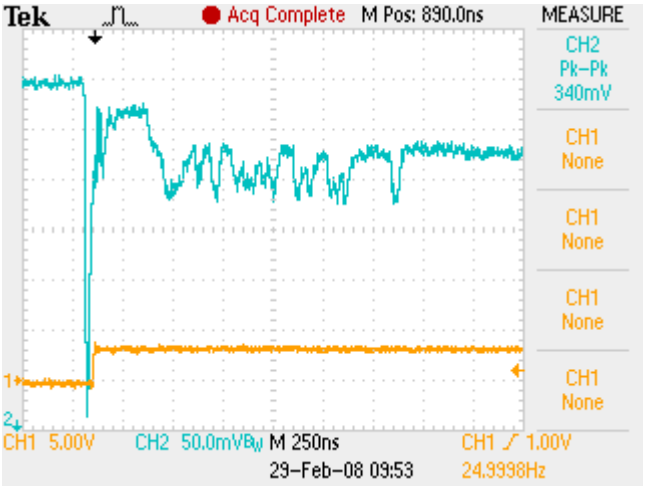

Figure 21 – Measured impact of Power Gating without "soft-start" turn on (SALT1/TSMC90G)

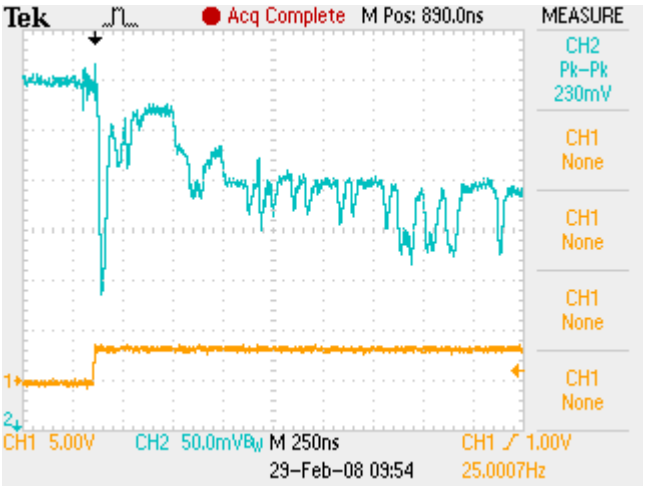

Figure 22 – Measured impact of Power Gating with "soft-start" turn on (SALT1/TSMC90G)

6.3. Addressing Power Gating testability

Testability has been addressed as follows

- Clamp signals are overridden in test mode for standard logic test
- Power gating controls are overridden for standard logic test
- Resets are made controllable to the tester
- The "ACK" acknowledge outputs from the power gating chains support basic buffer chain integrity checking

This is shown in **Figure 23**. Highlighted enhancements include the test mode control of the control output signals.

Standard ATPG testing is supported for the logic in operational (powered-on) mode, while special test bench sequences are used to extract specific test vector sets for the power gating and state retention control sequencing.

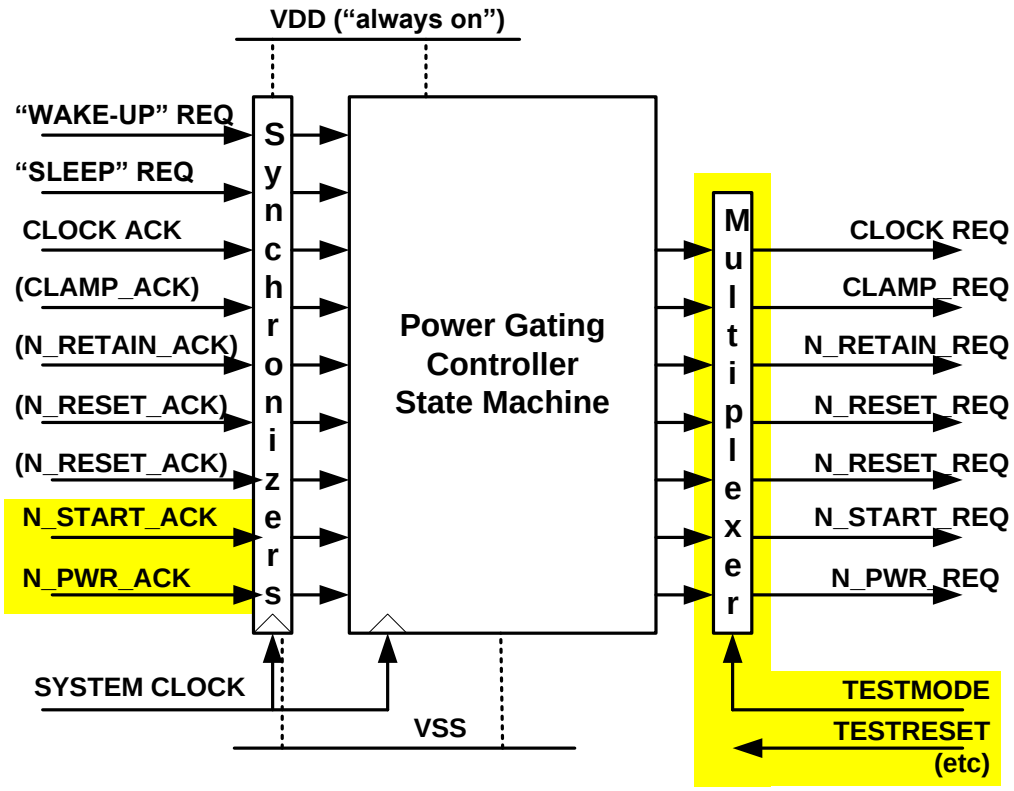

Figure 23: Power Gating State Machine with Testability support added

6.4. Addressing Power Gating Topology

A distributed power-switch mesh topology has been used more typically than a ring switch approach.

The power gates used can simply be viewed as transistor switch replacements for the power deliver via stacks that typically distribute power from thick upper metal layers (typically vertical power stripes) and provide (horizontal) metal-1 power delivery to the power-gated standard cells.

Extra vertical metal power straps are added to grid the power gated switched rails together for good inter-switch current sharing. The power gates with integrated buffering also derive their driver power from the un-switched power network.

The approach is amenable to providing un-switched power to retention registers and their control signal buffering.

Figure 24 illustrates the distributed power switch grid topology to show the control chain ports. Rather than have a single long “MAIN” power control switch chain (which may have thousands of switches) a number of parallel chains are implemented and these can then be externally driven by a “PWR_REQ” shared driver. The independent “PWR_ACK” signals do support a basic level of testability for all the chains.

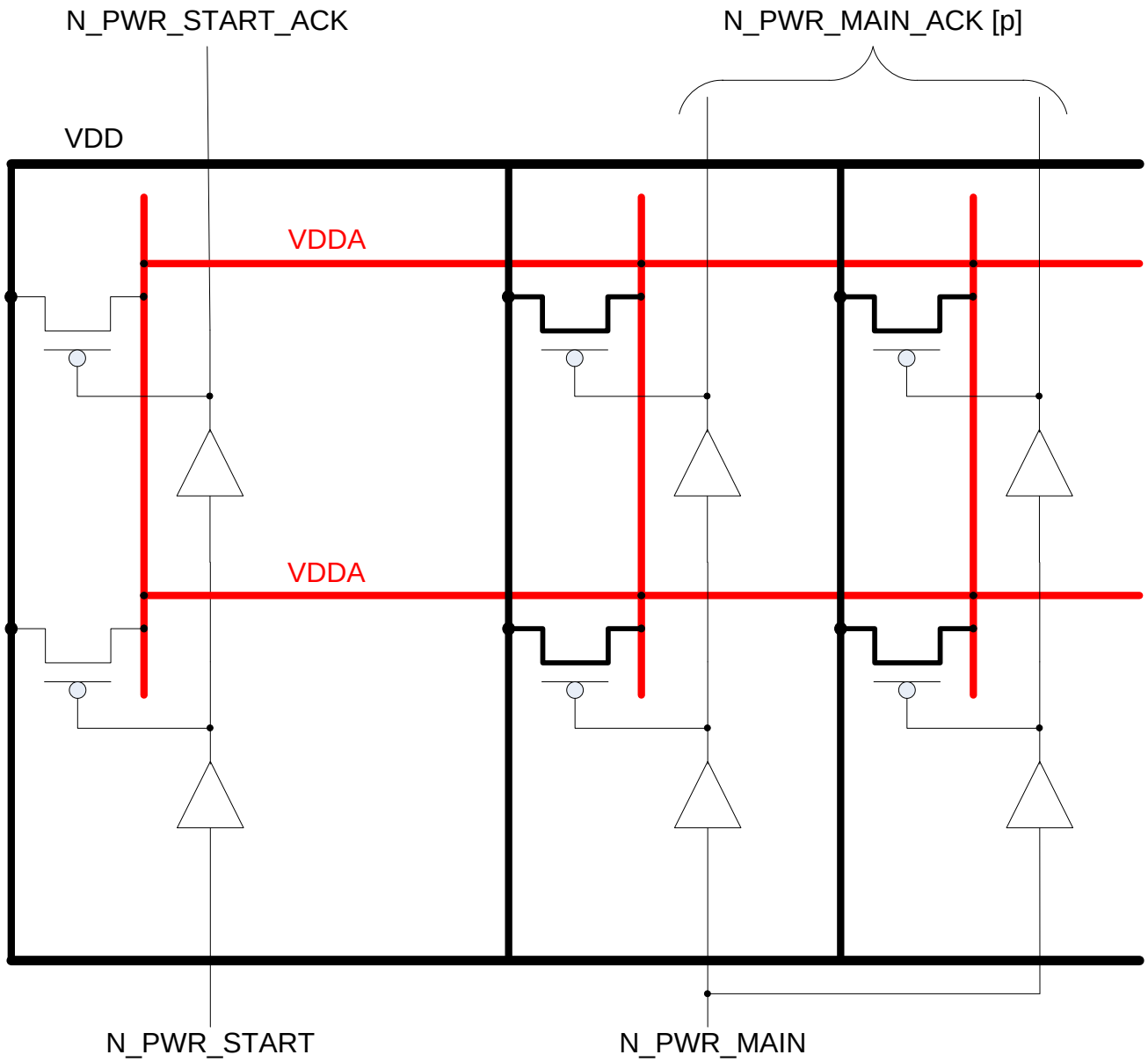

Figure 24: Power gating switch topology

7 Conclusions: best-practice – and UPF pitfalls for the unwary

Power rail switching of subsystems to cut leakage power has proven to be a valuable technique for reducing standby power in portable applications. The ability to overlay power-intent cleanly on an RTL design is now fully supported in the current UPF tools including the full suite of Synopsys synthesis, layout and verification tools now available.

It is well understood in the industry that imposing RTL design best practices can ensure IP is cleanly reusable across a range of process technologies and design/implementation/verification flows.

This paper has described the background principles and the more practical details of power management approaches for power gating with due attention to clocks, resets, testability and safe power sequencing – all based on fabricated and silicon proven technology demonstrators.

The UPF "pitfalls for the unwary" are summarized as:

- UPF can infer basic power gating, but design-time analysis is required to factor in appropriate power-up control strategies
- UPF infers basic interface output clamping in a straightforward manner but the designer need to specify the quiescent/safe values to be used
- How fast a power gated network can be turned-on requires attention both the UPF inferred power gating structures and the RTL controller design
- Designing in handshake protocols across the UPF and RTL divide can facilitate reuse and technology portability
- Power gating is not instantaneous or as transparent as clock-gating can be. Clocks cannot be restarted until power rails are safely stabilized and valid. Resets can either be handled explicitly or require careful asserting with UPF-inferred input clamping.

State Retention Power Gating needs special care.

- Arbitrary state retention inference through UPF is strongly to be discouraged
- For legacy IP, 'total state' or no state retention are the options that will verify and work
- All the above provided that state integrity is not compromised by the power-gating switch-on transients of not only the block in question – but neighboring blocks on the SOC that share a common ground or power rail
- Single-edge clocking is strongly advocated otherwise clock gating constructs will have to infer retention latches for functional clock gating.

7.1.1. Acknowledgements

In particular thanks are due to Alan Gibbons of Synopsys, and John Biggs, James Myers, Sachin Idgunji and Leah Schuth all with ARM for their practical support and commitment to the SALT technology demonstrator program – resulting in a series of right-first-time test chips.

8 References

[1] ITRS 2000: http://www.itrs.net/Links/2000UpdateFinal/ORTC2000final.pdf

[2] Kim N., Austin T., Blaauw D., Mudge T., Flautner K., Hu J., Irwin M., Kandemir M., Narayanan V., "Leakage current: Moore's law meets static power" IEEE Computer Vol. 36, Issue 12, 2003

[3] Mudge, Trevor, "Power: A First Class Architectural Design Constraint" IEEE Computer, vol. 34, no. 4, April 2001. http://doi.ieeecomputersociety.org/10.1109/2.917539

[4] Accellera UPF Standard version 1.0, February 2007. http://www.accellera.org/apps/group_public/download.php/989/upf.v1.0.pdf

[5] Mutoh S. et al. "A 1v multi-threshold voltage CMOS DSP with an efficient power management technique for mobile phone applications" ISSCC1996, pages 168–169, 1996.

[6] Biggs, J. Gibbons, A. "Aggressive Leakage Management in ARM-based Systems" SNUG Boston 2006

[7] Flynn, D.. Gibbons, A. "Aggressive Leakage Mitigation in ARM Processor Based Systems" Tutorial MC4, SNUG San Jose 2007 http://www.synopsys.com/news/pubs/snug/sanjose07/mc4_tutorial.pdf

[8] Keating M., Bricaud P., "Reuse methodology manual: for system-on-a-chip designs" Kluwer Academic Publishers, 1998. ISBN:0-7923-8175-0

[9] Flynn, D.. Gibbons, A. "Design for State Retention: Strategies and Case Studies" SNUG San Jose 2008, Track TA2

8.1.1. Bibliography

- Keating M., Bricaud P., "Reuse methodology manual: for system-on-a-chip designs" Kluwer Academic Publishers, 1998. ISBN:0-7923-8175-0
- Chandrakasan A., Sheng, Brodersen R., "Low-Power CMOS Digital Design", IEEE JSSC, April 1992 p473-484
- Chandrakasan A., Brodersen R., "Low-Power CMOS Design", IEEE Press 1998, ISBN 0-7803-3429-9
- Massoud Pedram , Jan M. Rabaey, Power Aware Design Methodologies, Kluwer Academic Publishers, Norwell, MA, 2002. ISBN:1402071523
- Keating M., Flynn D., Aitken R., Gibbons A., Shi K., "Low Power Methodology Manual for System-on-Chip Design", Springer 2007, ISBN 978-0-387-71818-7