

# Power Intent and Low Power Methodology

Dr. Adam Teman

24 June 2020

# Outline



## Standard Low Power Methods



## Multi-Supply Voltage (MSV)



## Power Gating



## CPF

Standard  
Methods

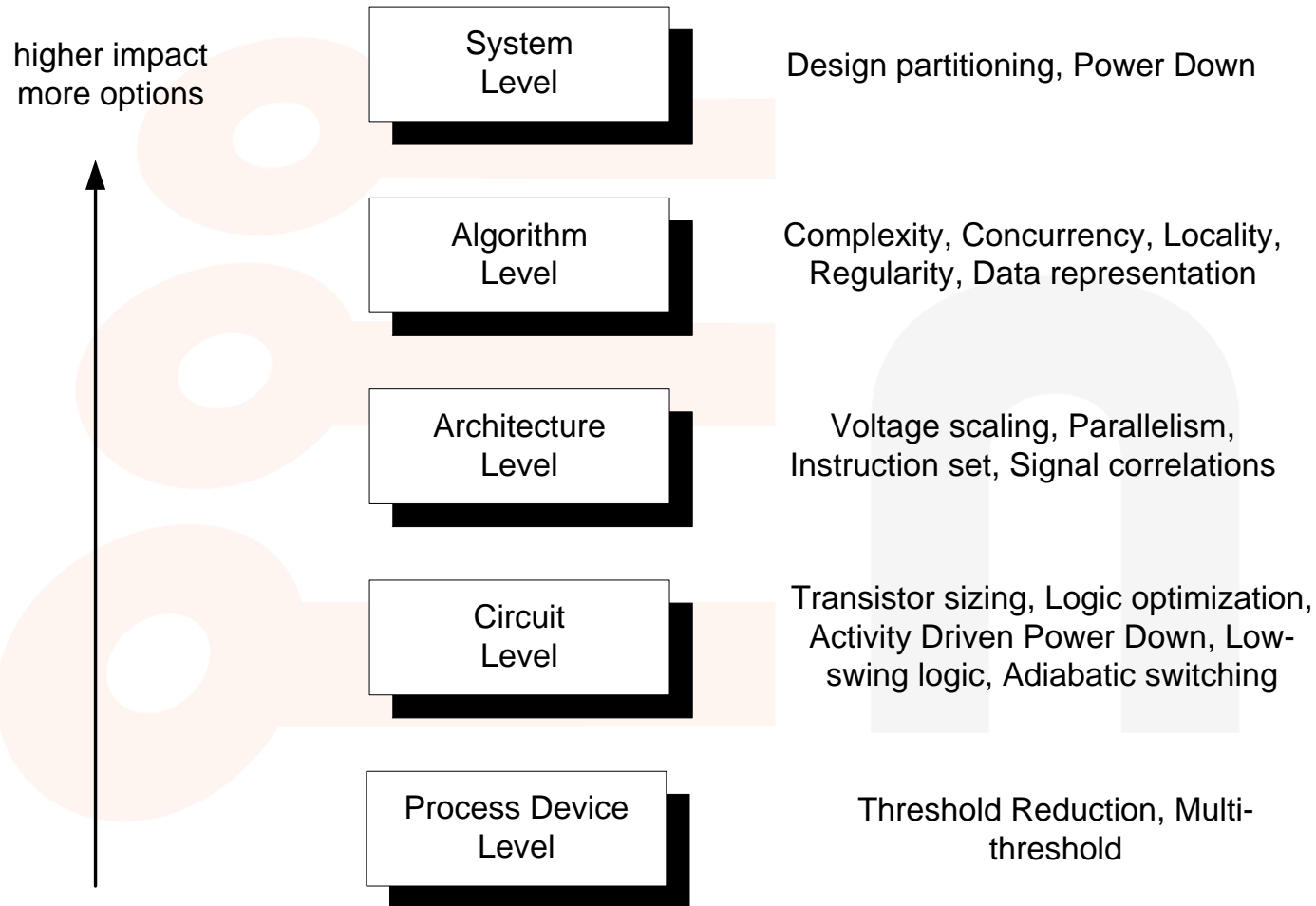
MSV

Power  
Gating

CPF

# Standard Low Power Methods

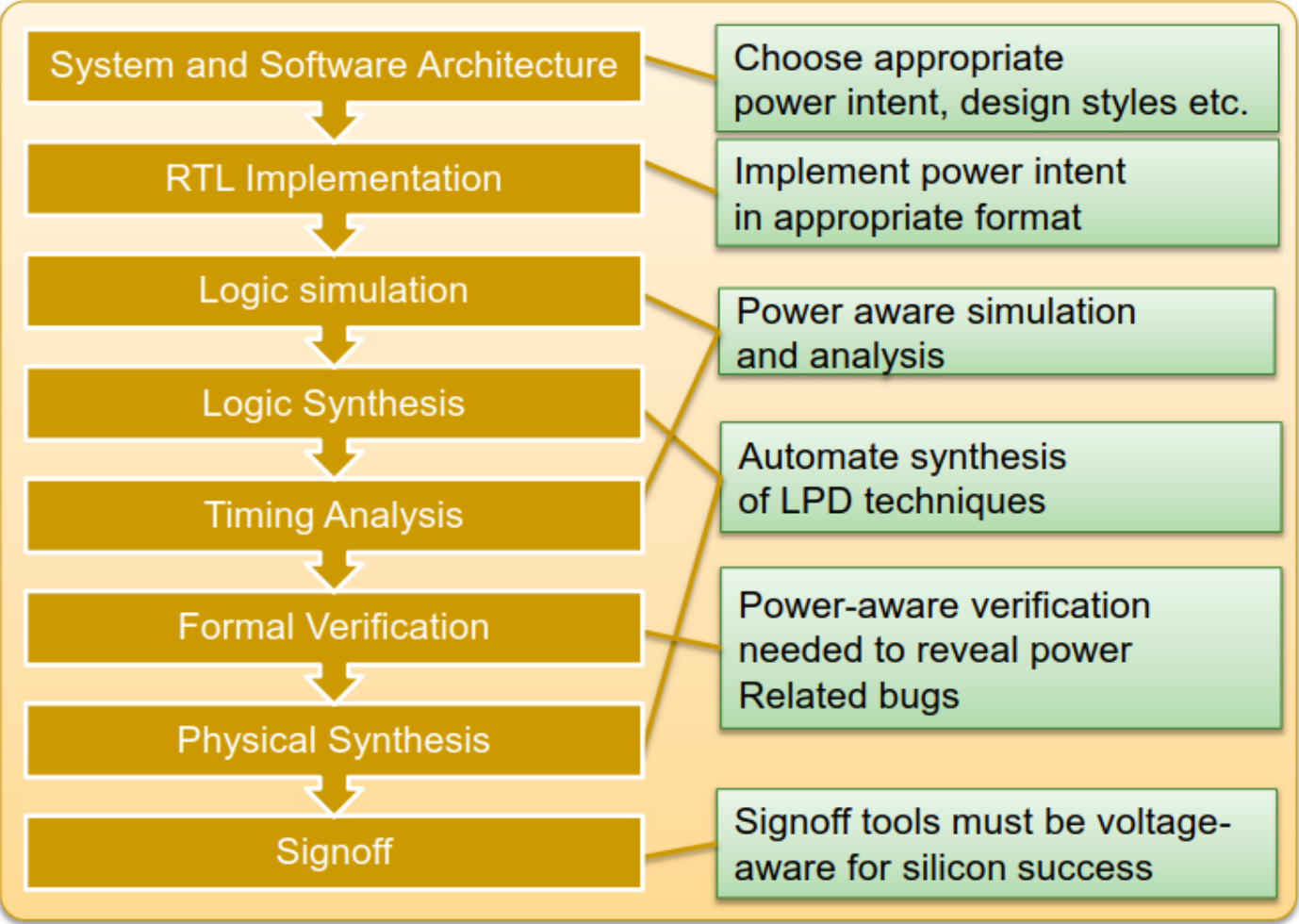
# Low Power at various levels of circuit design



Source: U. Nis

Algorithmic	Algorithm selection	orders of magnitude
Behavioral	Concurrency Memory	several times
Power manage	Clock ctrl	10-90%
RT Level	Structural transform.	10-15%
Tech. indepen.	Extraction/decomp.	15%
Tech dep.	Tech. mapping Gate sizing	20% 20%
Layout	Placement	20%

# Power-Aware Design Flow

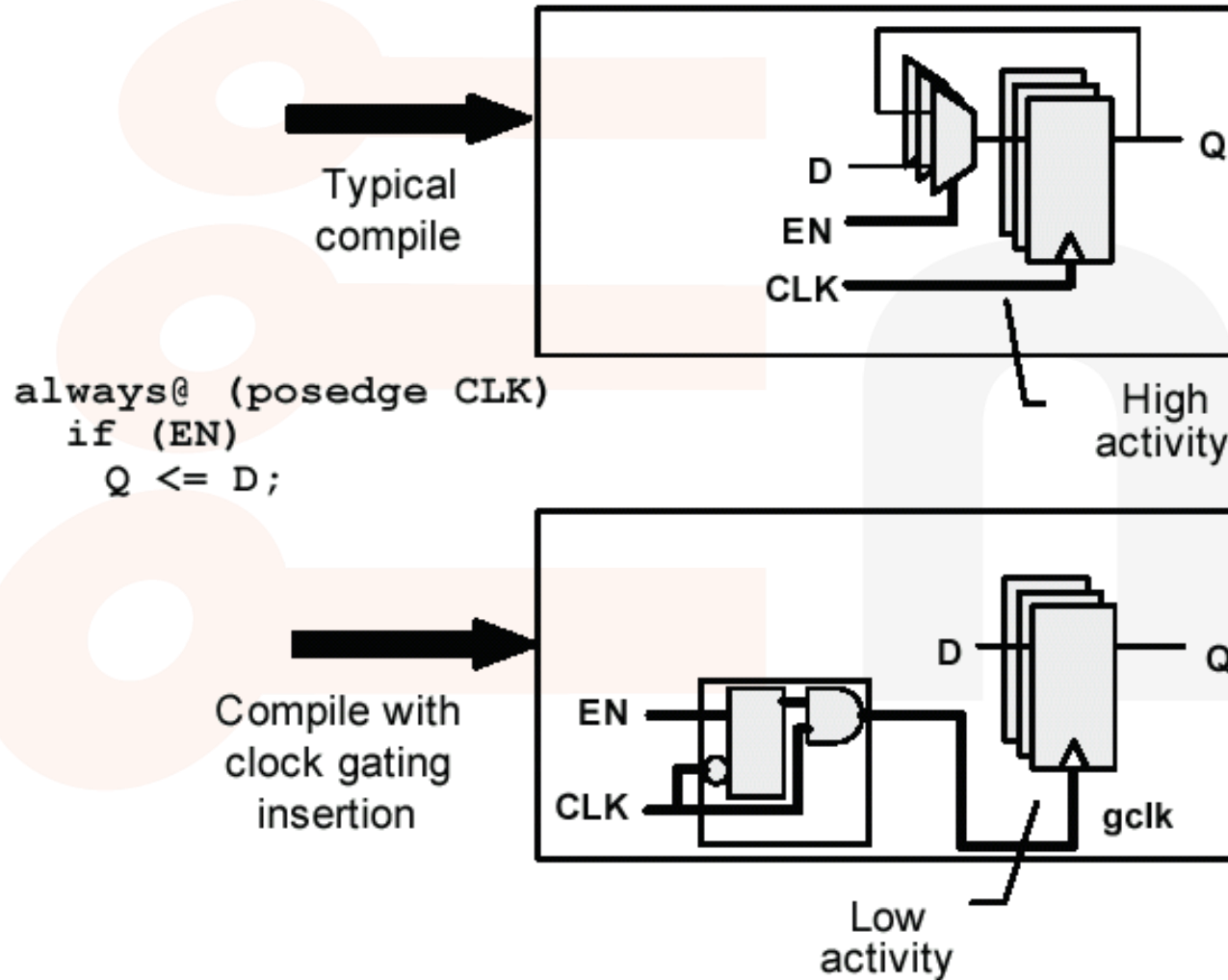


Source: Synopsys

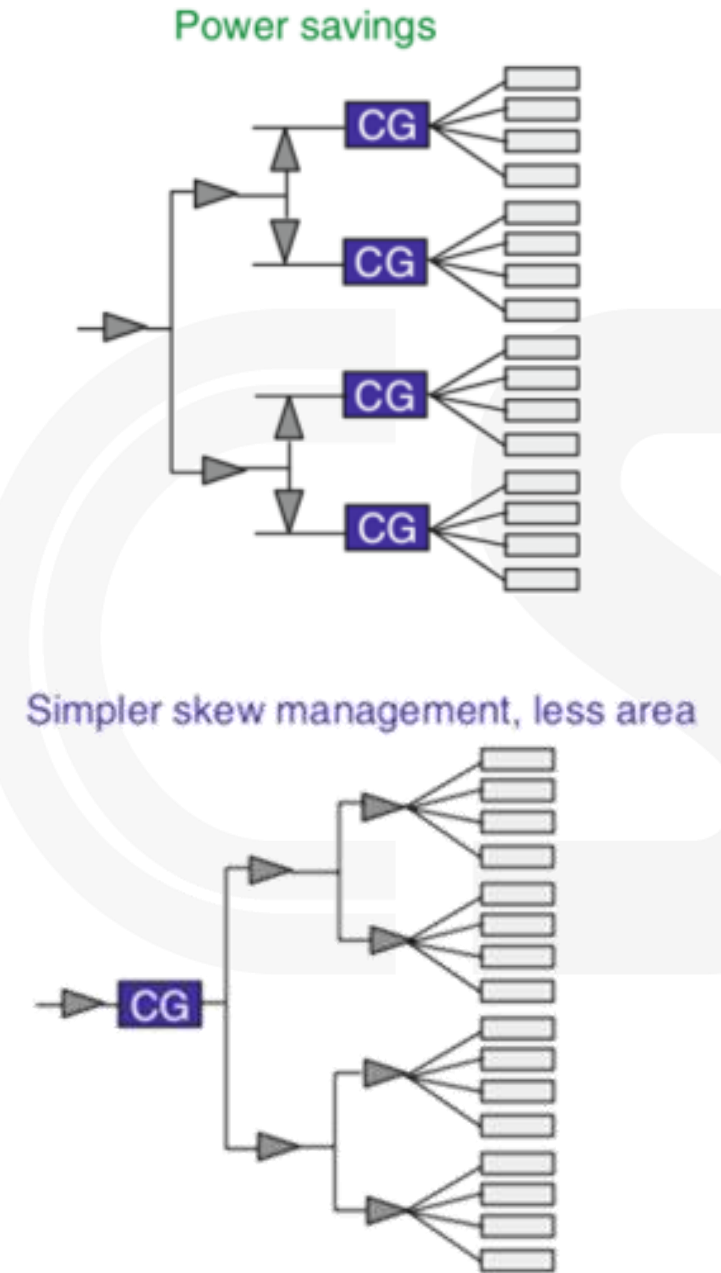
Libraries	RTL Power Constructs	Definition of power domain Isolation behavior of a particular signal Retention behavior of particular registers
	RTL Simulation	Power domain simulation Isolation logic simulation
	Logic Synthesis	Create Power Domains Clock Gating Apply OpCond on blocks Special cell Insertion Retention Cell Synthesis Compile MV DFT
	Physical Implementation	Voltage Area Creation MTCMOS Insertion Physical synthesis Leakage optimization MCMM Scan reordering MV aware CTS MV aware Routing
	Verification	RTL vs. Gates matching Static Low Power Checks
	Signoff	Parasitic Extraction
		SI, Timing, Power Signoff
		Power Network Analysis

Source: Synopsys

# Clock Gating



Source: Keating

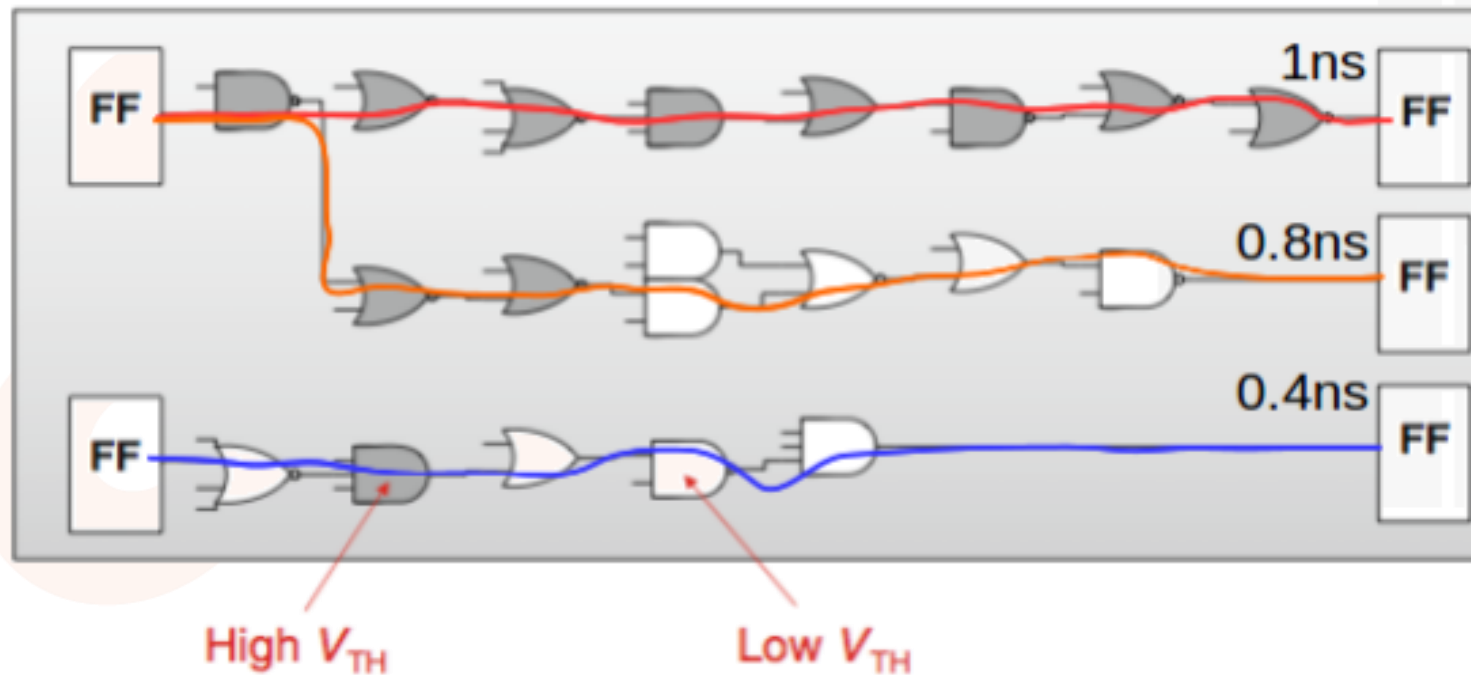


- Gate remapping
- Cell sizing
- Buffer insertion

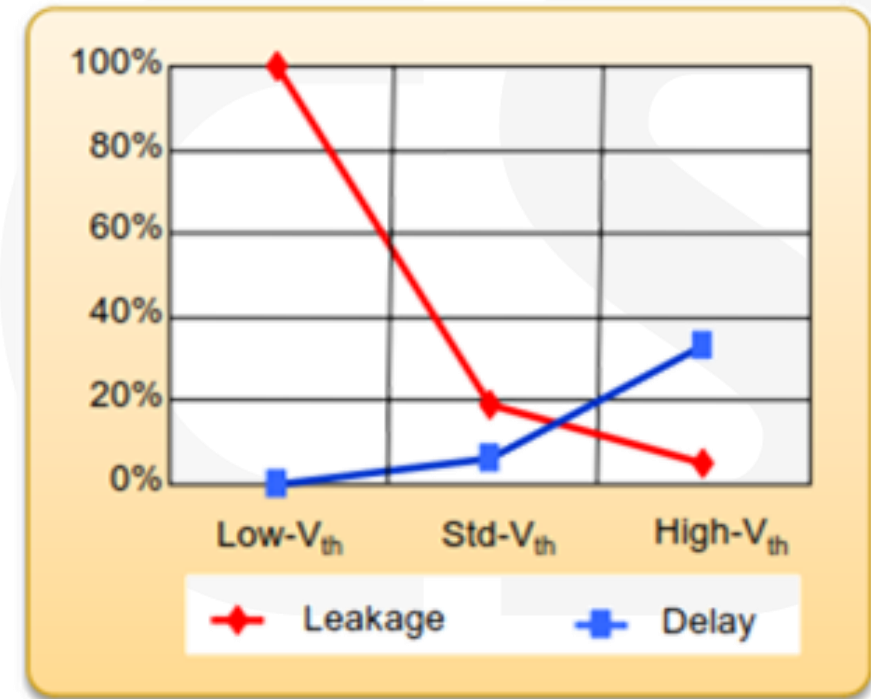


# Multi-Threshold Logic

Target Delay: 1ns  
Target Clock Frequency: 1GHz



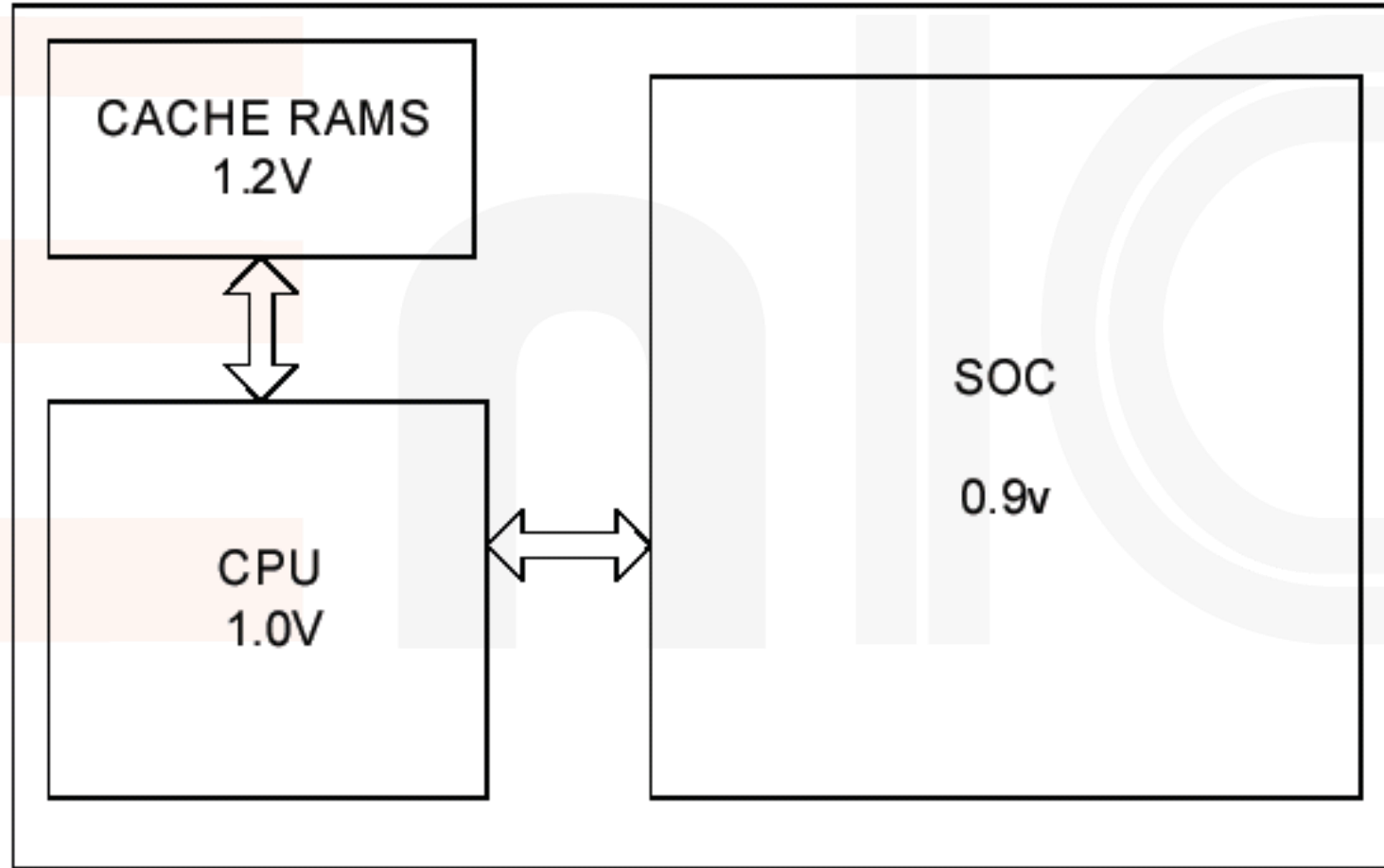
Source: Rabaey



Source: Synopsys

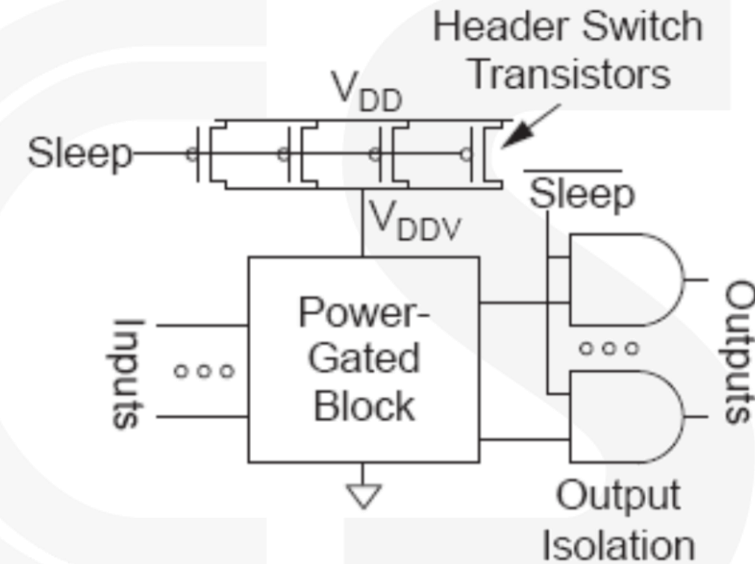


# Multi VDD

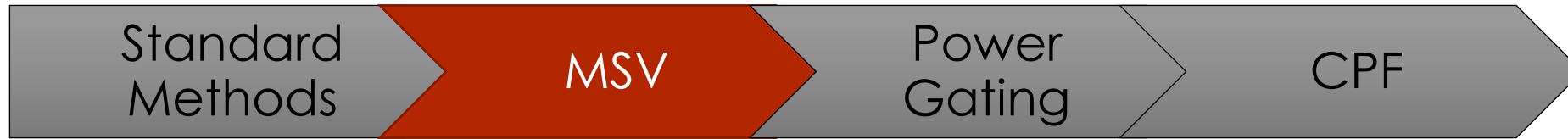


# Power Gating

- **Turn OFF power to blocks when they are idle to save leakage**
  - Use virtual  $V_{DD}$  ( $V_{DDV}$ )
  - Gate outputs to prevent invalid logic levels to next block
- **Voltage drop across sleep transistor degrades performance during normal operation**
  - Size the transistor wide enough to minimize impact
- **Switching wide sleep transistor costs dynamic power**
  - Only justified when circuit sleeps long enough



Source: CMOS VLSI Design



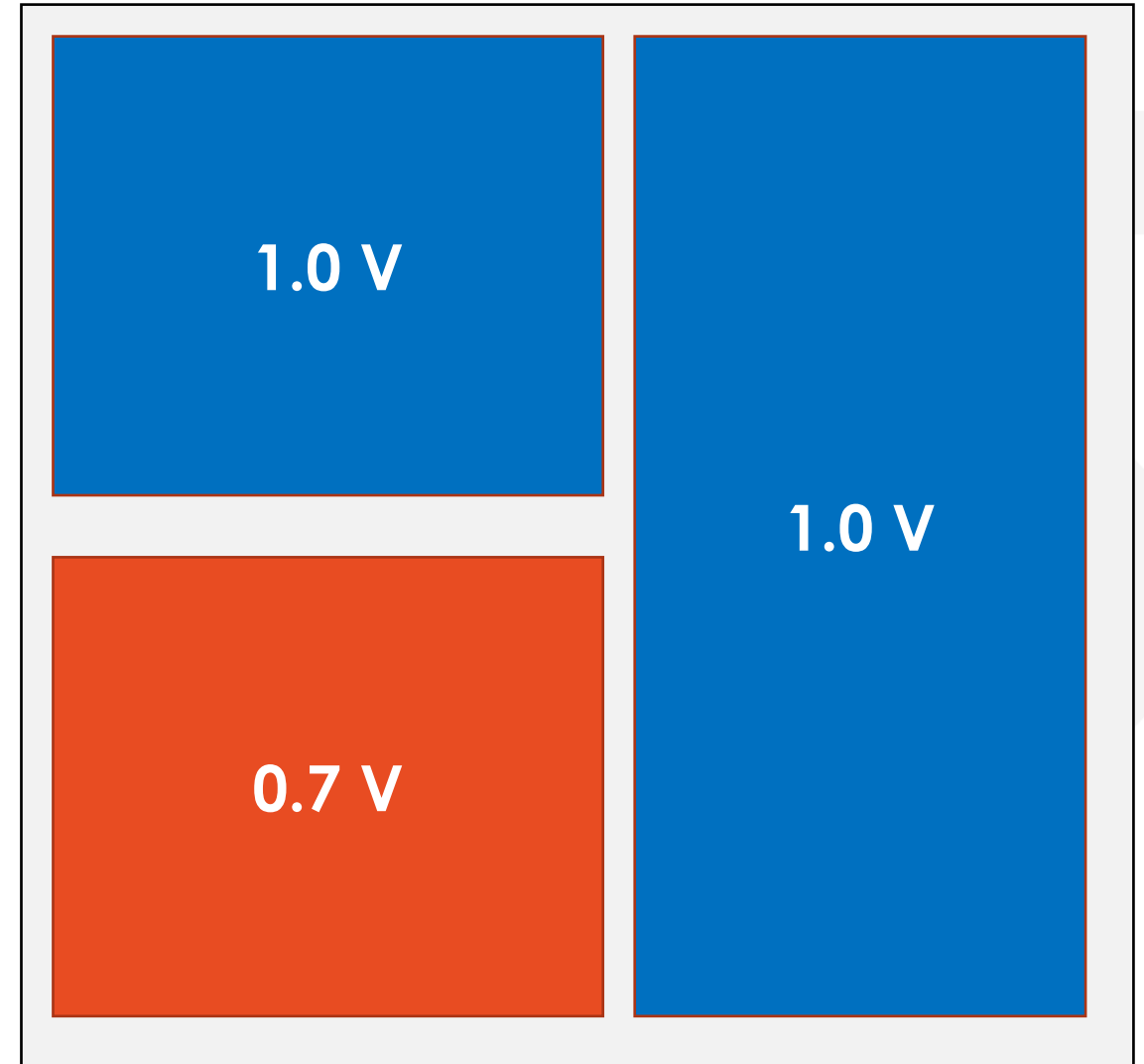
# Multi-Supply Voltage (MSV)

# Multi-voltage Strategies

- **Static Voltage Scaling (SVS):**
  - Different blocks or subsystems are given different, fixed supply voltages.
- **Multi-level Voltage Scaling (MVS):**
  - A block or subsystem is switched between two or more voltage levels.
  - Only a few, fixed, discrete levels are supported for different operating modes.
- **Dynamic Voltage and Frequency Scaling (DVFS):**
  - A larger number of voltage levels are dynamically switched to follow changing workloads.
- **Adaptive Voltage Scaling (AVS):**
  - An extension of DVFS where a control loop is used to adjust the voltage.

# Static Multi-Voltage (MSV)

- Different parts of the chip running at different voltages
- Reduces dynamic power without affecting overall performance
- Requires additional power supplies
- Requires level-shifter for cross-domain communication



# Multi-Voltage Design Challenges

- **Level shifters:**
  - Signals that go between blocks that use different power rails require level shifters.
- **Characterization and STA:**
  - Need libraries for each voltage and level shifter configuration.
- **Floorplanning:**
  - Floorplanning power domains
  - Complex power planning and power grids
- **Board level issues:**
  - Need additional regulators to provide the additional supplies.
- **Power up and power down sequencing:**
  - There may be a required sequence for powering up the design in order to avoid deadlock.

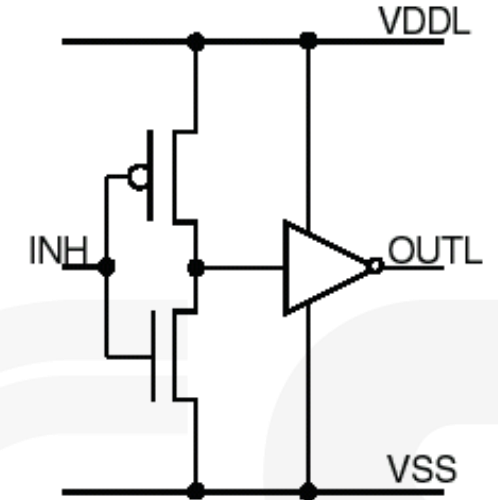
# High-to-Low Shifters

- **Why is a shifter required?**

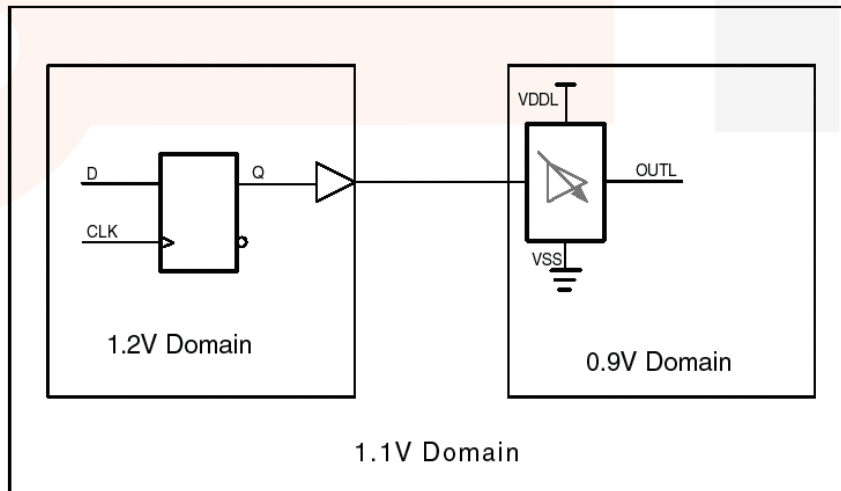
- You just need an inverter (or two)
- To provide characterization for a particular shift.

- **Where do you place it?**

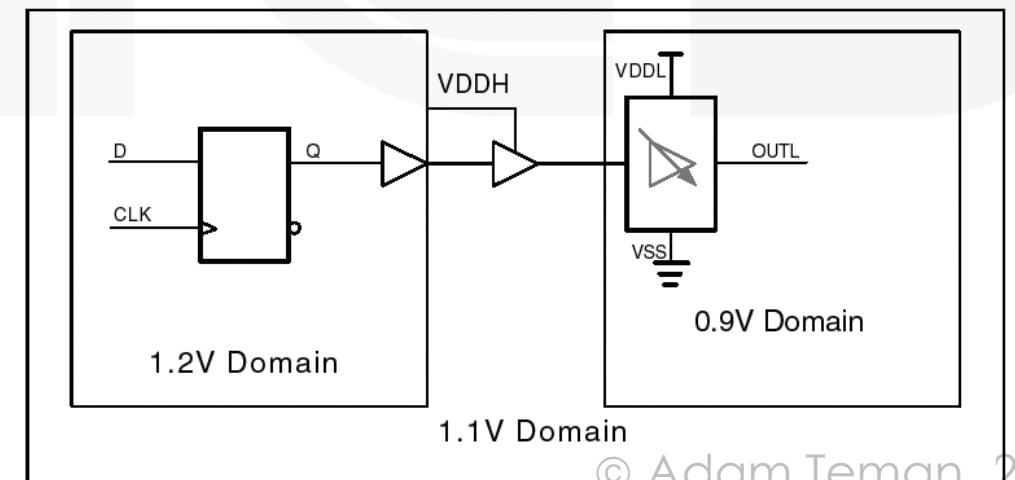
- Since it uses the low voltage, place it in the destination domain.
- But, if it is far from the high domain, buffering may be required.
- You can route the high voltage as a signal to the buffer.



Source: Keating

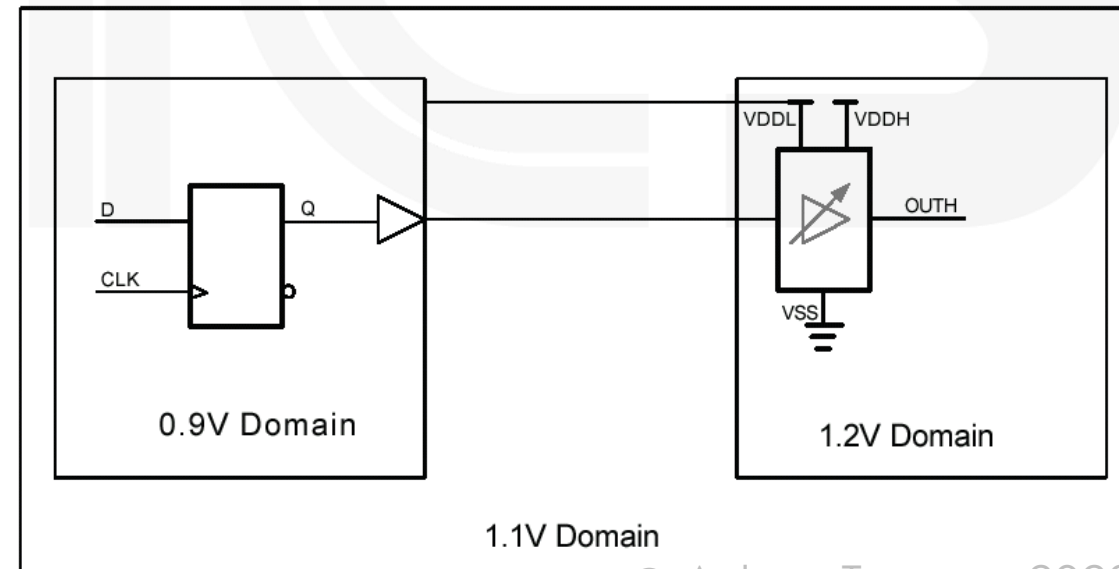
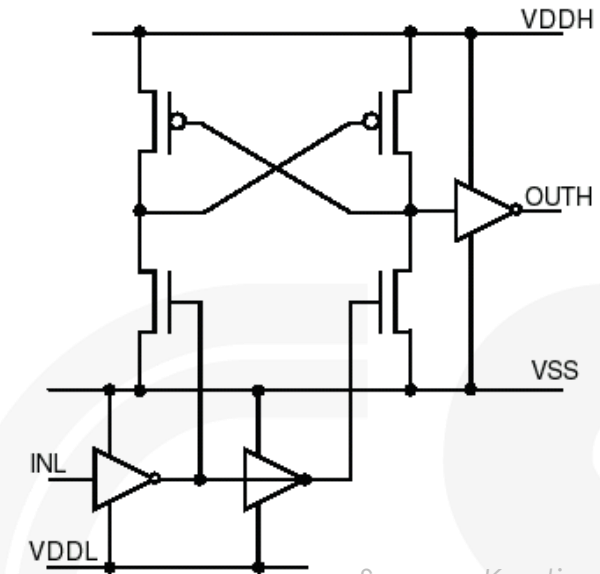


**Level Shifter in the Destination Domain**



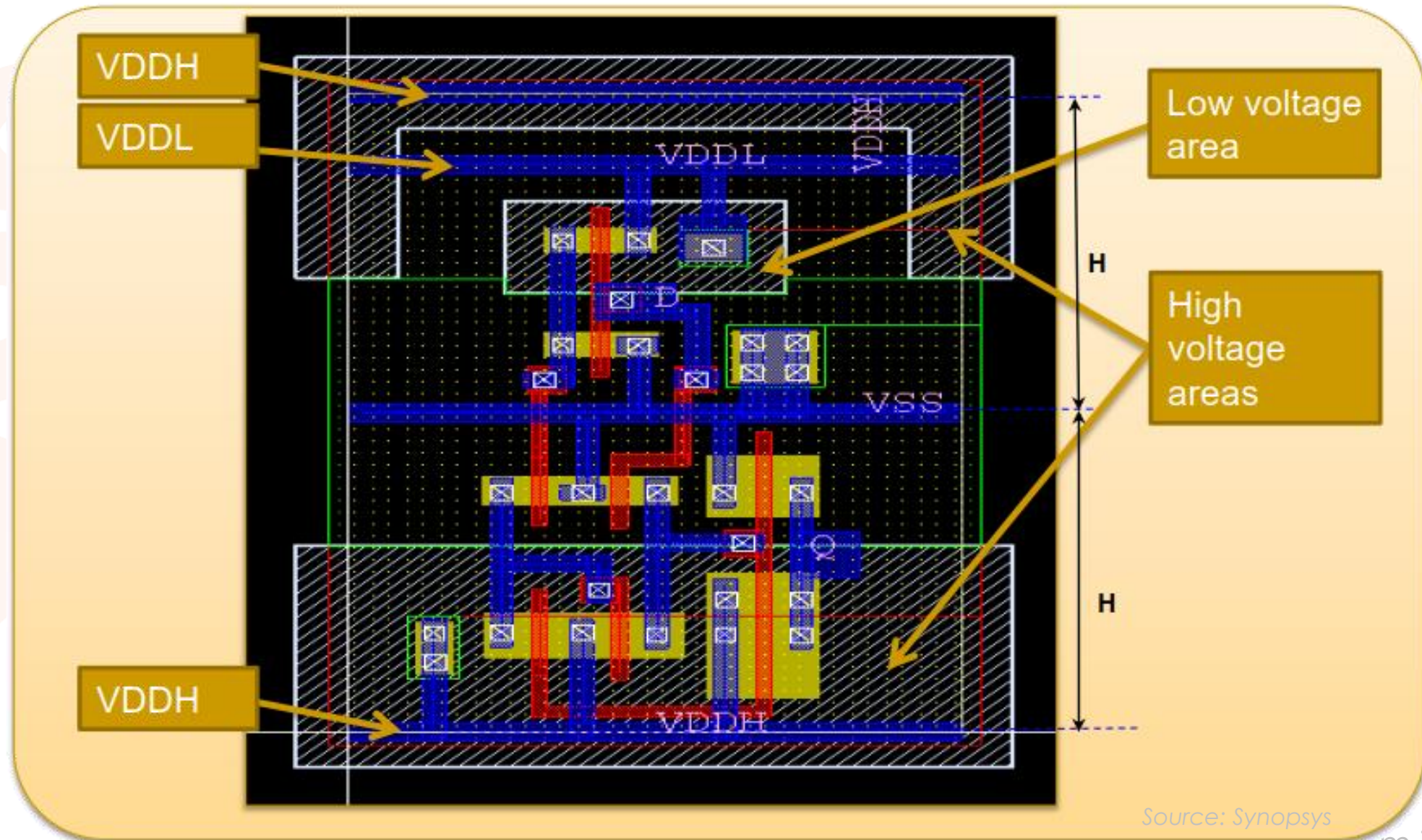
# Low to High Shifters

- **Why is it needed?**
  - Need to cross a threshold
  - Need to eliminate crowbar current
- **Shifter placement**
  - Always requires routing one voltage to the other domain as a signal.
  - Place in the destination domain, since the output driver requires more current.
  - Additional buffers required in main domain if distance is too large.





# Low-to-High Shifter Layout



# Level Shifters in EDA

- Can be specified in RTL or automatically added during floorplanning.
- CPF/UPF enable defining level-shifter rules:
  - When to add them (what voltage difference)
  - Placement (source or destination domains)
  - etc.
- **Recommendations**
  - Place in the receiving domain
  - Factor in the delay of low-to-high shifters in the RTL partitioning for timing
  - Ensure that the voltage relationship between domains is clear so the tools can insert the right type of shifter.
  - If bi-directional shifting is required, setup and hold timing becomes complex.

# Timing and Power Planning in MSV

- **STA**

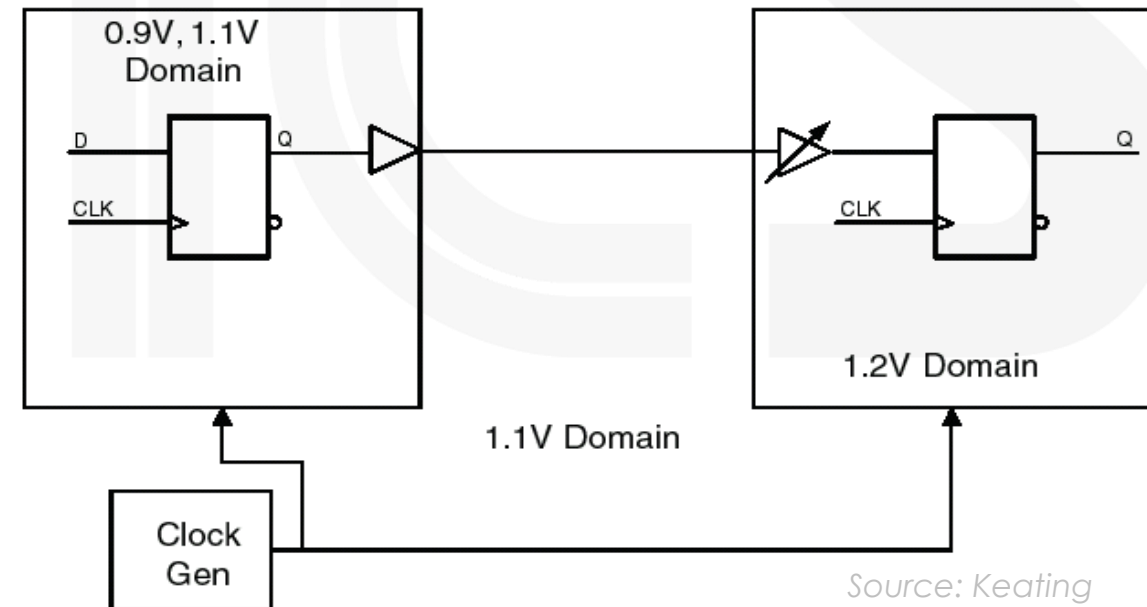
- Characterization for all voltages and shifting possibilities must be provided.
- Each mode must be defined with its own constraints.
- Optimization must be carried out simultaneously (MMMC)

- **CTS**

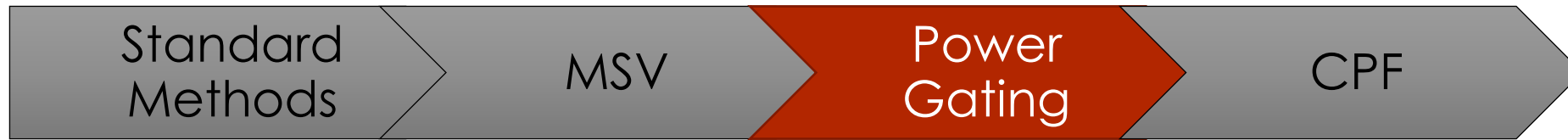
- CTS needs to understand level shifters
- If multiple voltages are supported, skew minimization has to be done for both modes.

- **Power Planning**

- Each domain has to be provided with its own voltage and power grid.



Source: Keating



# Power Gating



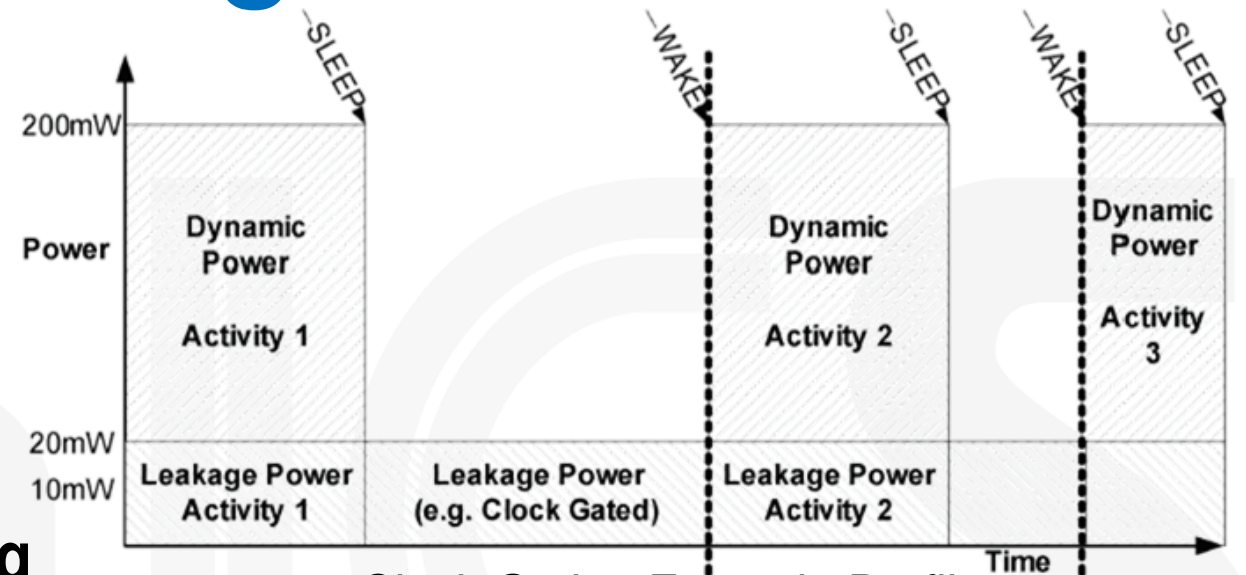
# Trade-offs in Power Gating

- **Clock Gating vs. Power Gating**

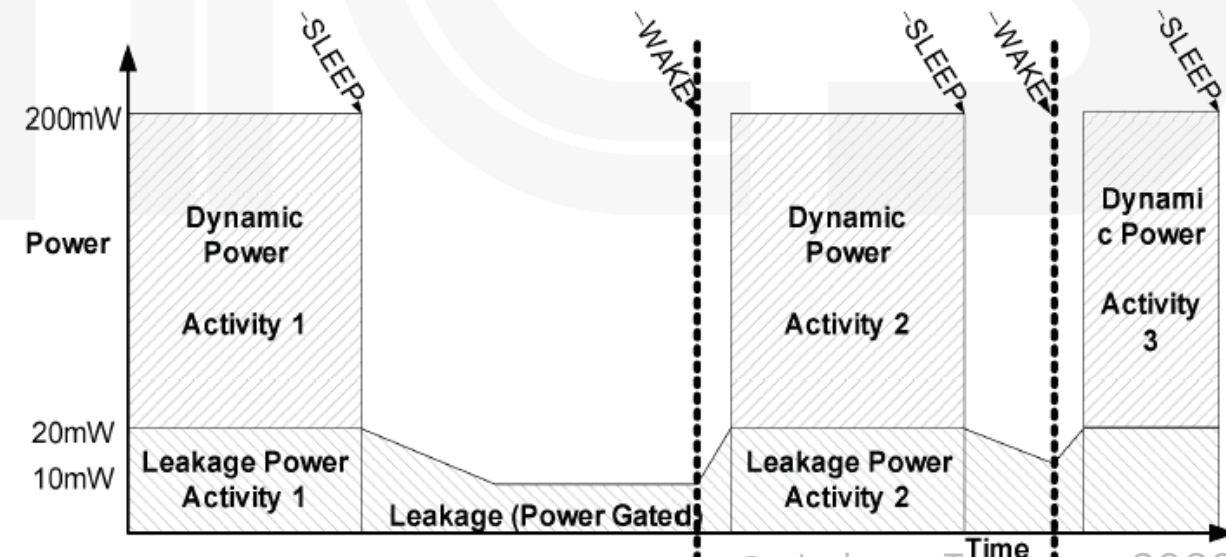
- Clock Gating is immediate, while power gating takes time
- Leakage under power gating is much lower than under clock gating

- **Architectural Tradeoffs for Power Gating**

- Possible savings in leakage power
- Entry and exit time penalties
- Energy required to enter and leave such leakage saving modes
- The activity profile (proportion and frequency of times asleep or active)



Clock Gating Example Profile *Source: Keating*



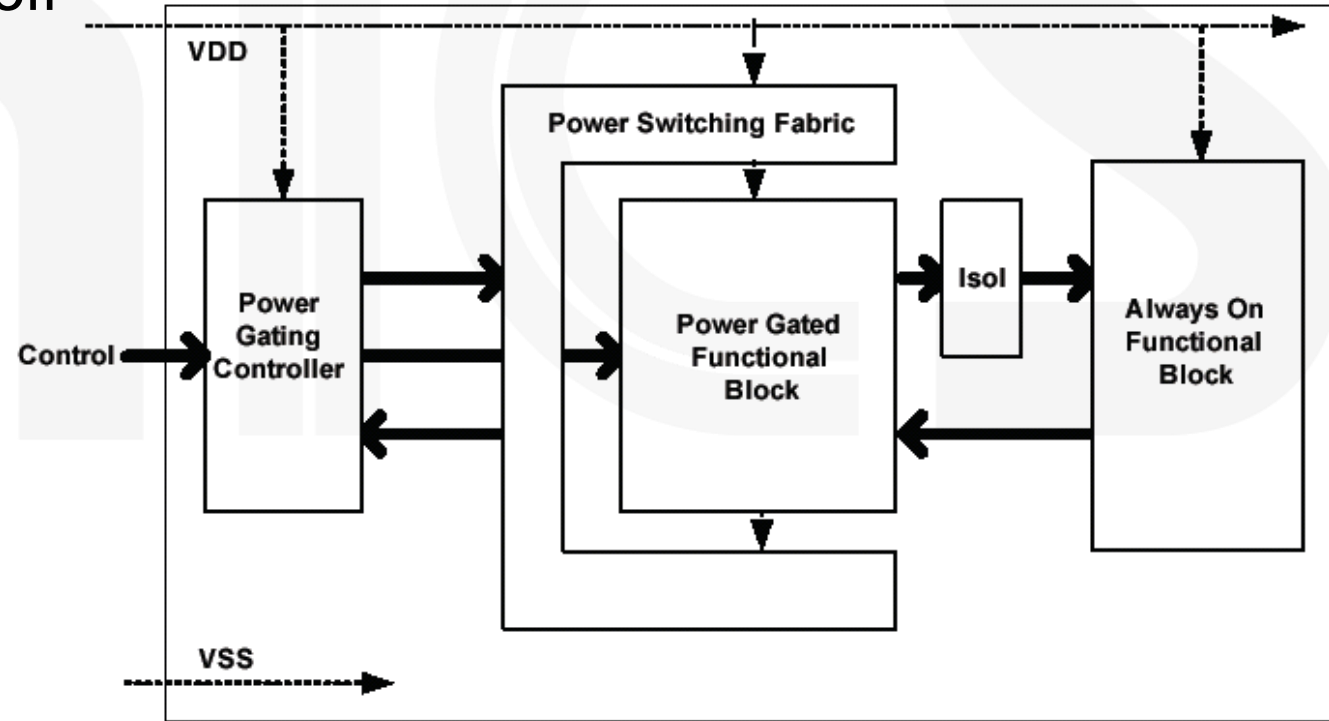
Power Gating Example Profile *© Alan Teman, 2020*

# Power Gating a CPU

- **A cached CPU may be inactive for long periods making power gating attractive**
- **However:**
  - Wake up time in response to an interrupt may be too long
  - If cache contents are lost every time the CPU is powered down then there is likely to be a significant time and energy cost in all the bus activity to refill the cache when it is powered up.
  - The net energy savings depend on the sleep/wake activity profile as to how much energy was saved when power gated versus the energy spent in reloading state.
- **For multi-processor systems, power gating is very advantageous**
  - Power gate once a CPU has finished its task → cache contents are no longer needed
- **Some peripheral subsystems may also be very compatible with power gating**
  - Device drivers can be profiled and the operating system management scheme can be optimized.
  - Some internal state may need to be stored during sleep mode

# Principles of Power Gating

- A simplified view of a power-gated SoC:
  - VDD is supplied through a power switching network
  - VSS is supplied to the entire chip
  - A controller switches the block on/off
  - Isolation cells eliminate crowbar currents during power down
  - Retention registers can be used to retain state with low leakage during power down



Source: Keating

© Adam Teman, 2020

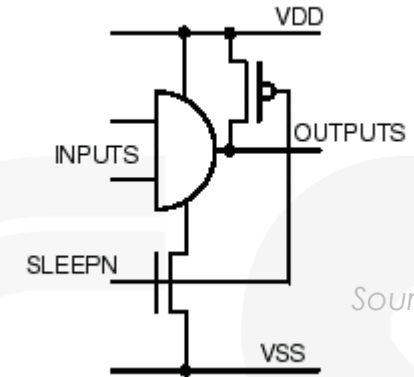
# Fine vs. Coarse Grain Power Gating

- **Fine Grain PG:**

- Local switch in each standard cell
- Huge area overhead (2x-4x)
- Standard flow for characterization and timing.

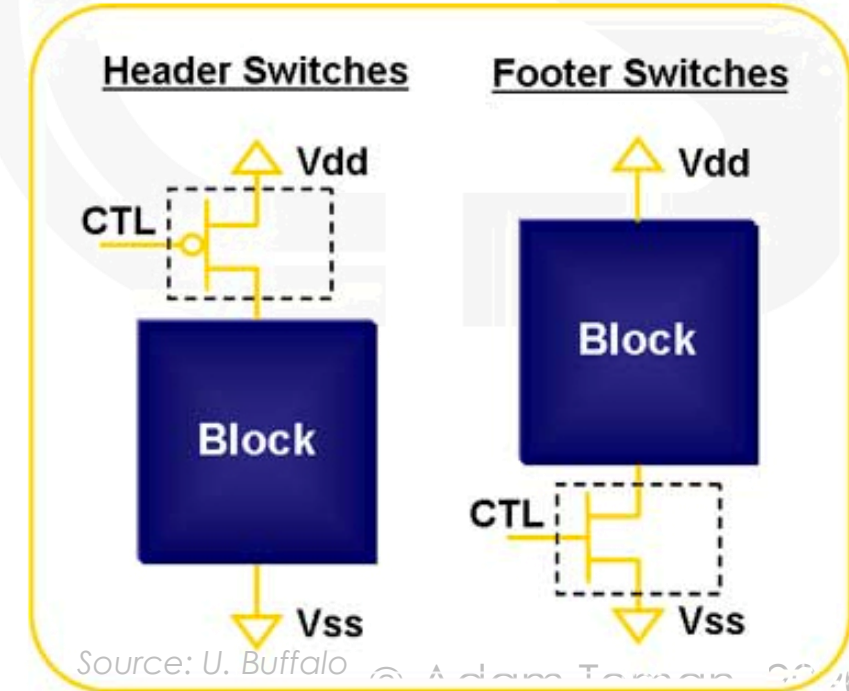
- **Coarse Grain PG:**

- Global (distributed) switch for a block of cells.
  - Much lower area overhead.
  - Very hard sizing estimation.
- **Virtually all power gated designs today employ coarse grain gating.**



Source: Keating

Fine Grain AND Gate with Pull-Up



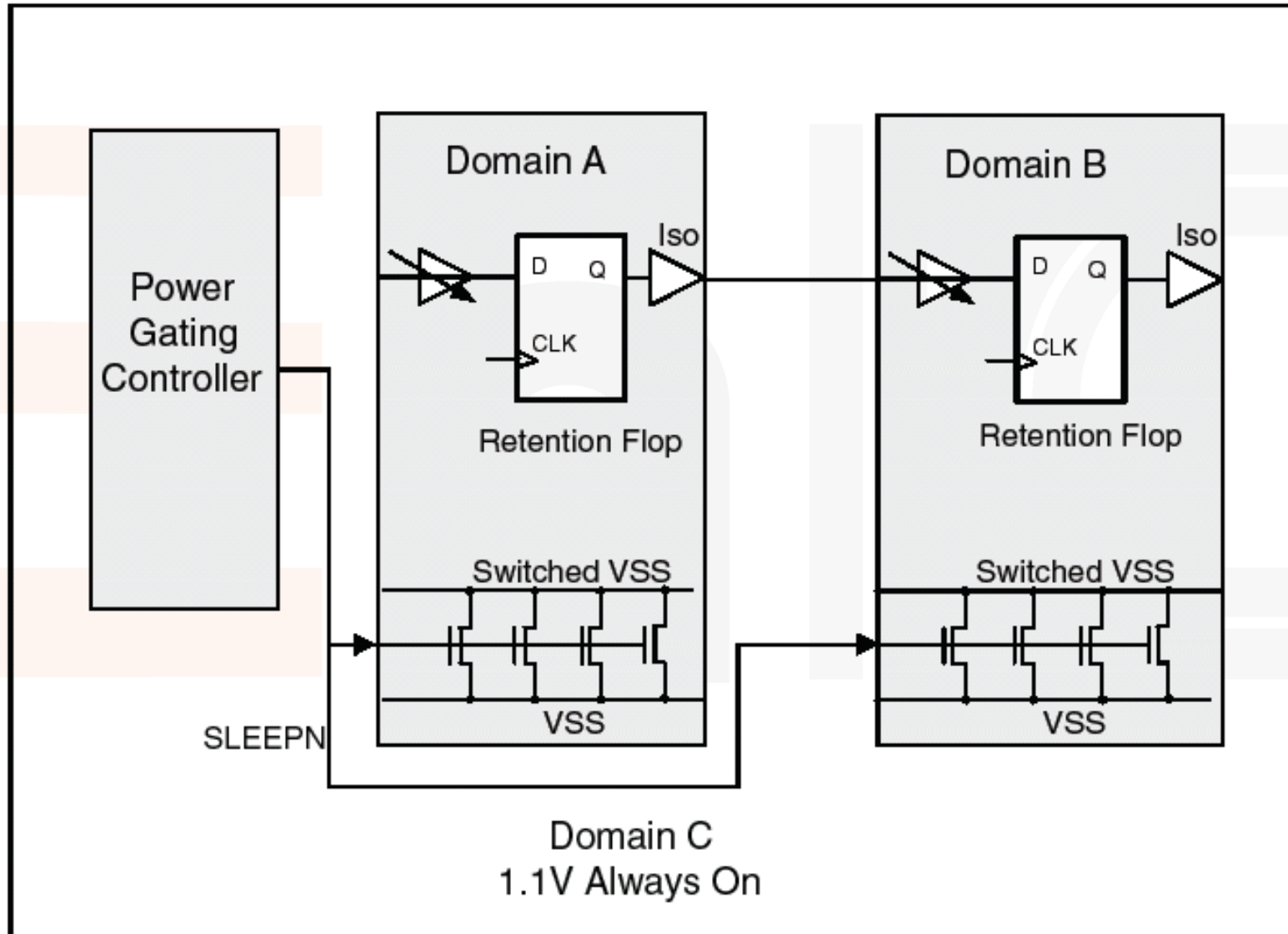
Source: U. Buffalo © Adam Tornay, 2020



# Power Gating Challenges

- Managing the in-rush current when the power is reconnected
- Design of the power switching fabric and power gating controller
- Selection and use of retention registers and isolation cells
- Minimizing the impact of power gating on timing and area.
- The functional control of clocks and resets
- Interface isolation
- Developing the correct constraints for implementation and analysis
- Verification for each supported power state and state transition
- Developing a strategy for manufacturing and production test

# Example Power Gated Design



# Power Gate Switches

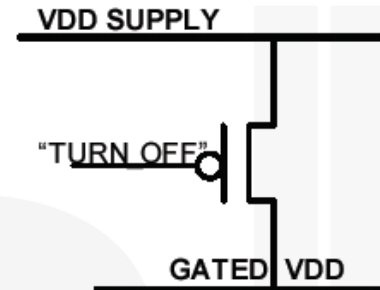
- **Switch Type:**

- “Header” (VDD switch)
- “Footer” (VSS switch)

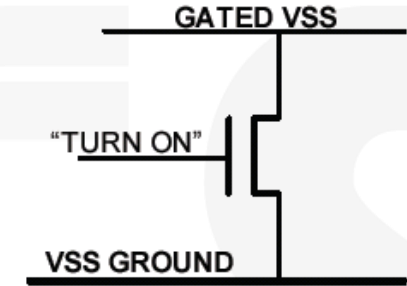
- **Tradeoffs**

- Using both headers and footers usually is unacceptable in terms of IR drop.
- Header switches are more popular, since external (off-chip) power gating can only be done with a header (VSS is common for the whole board)
- Level shifters usually share a common ground, causing footer switching to be a problem.
- It's easier for a designer to think of “off” as a signal falling to GND.

Basic Header-Switch structure:

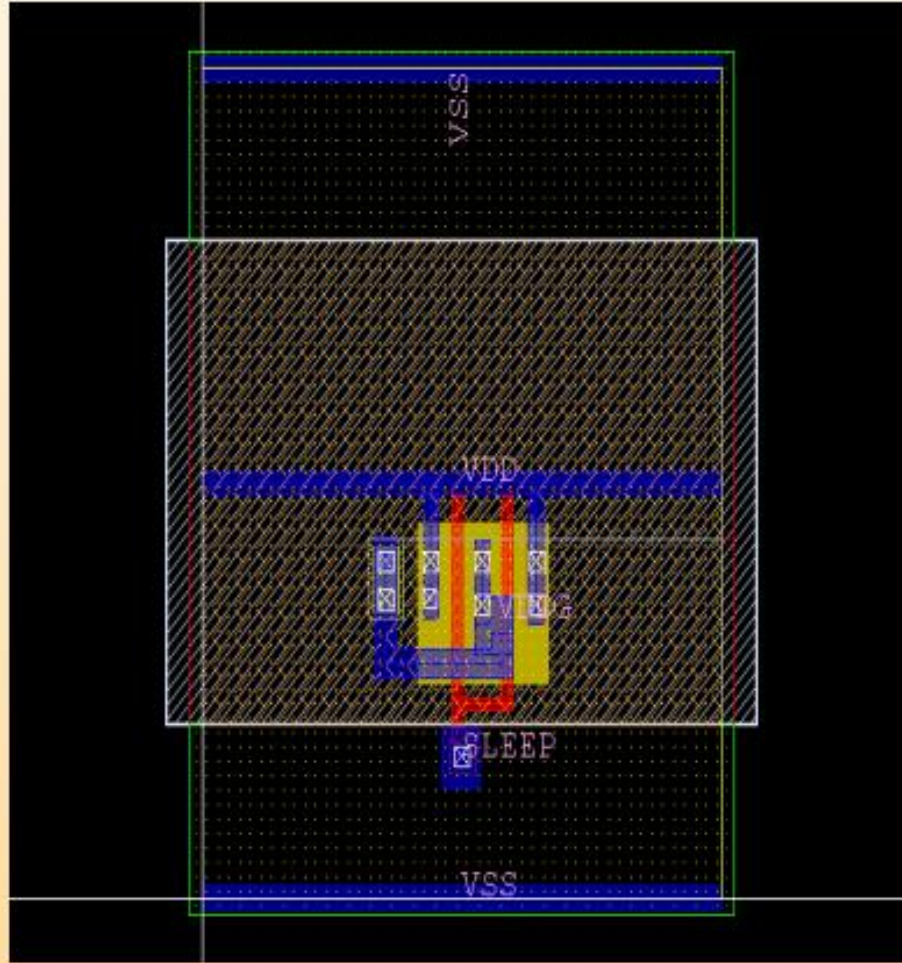


Basic Footer-Switch structure:

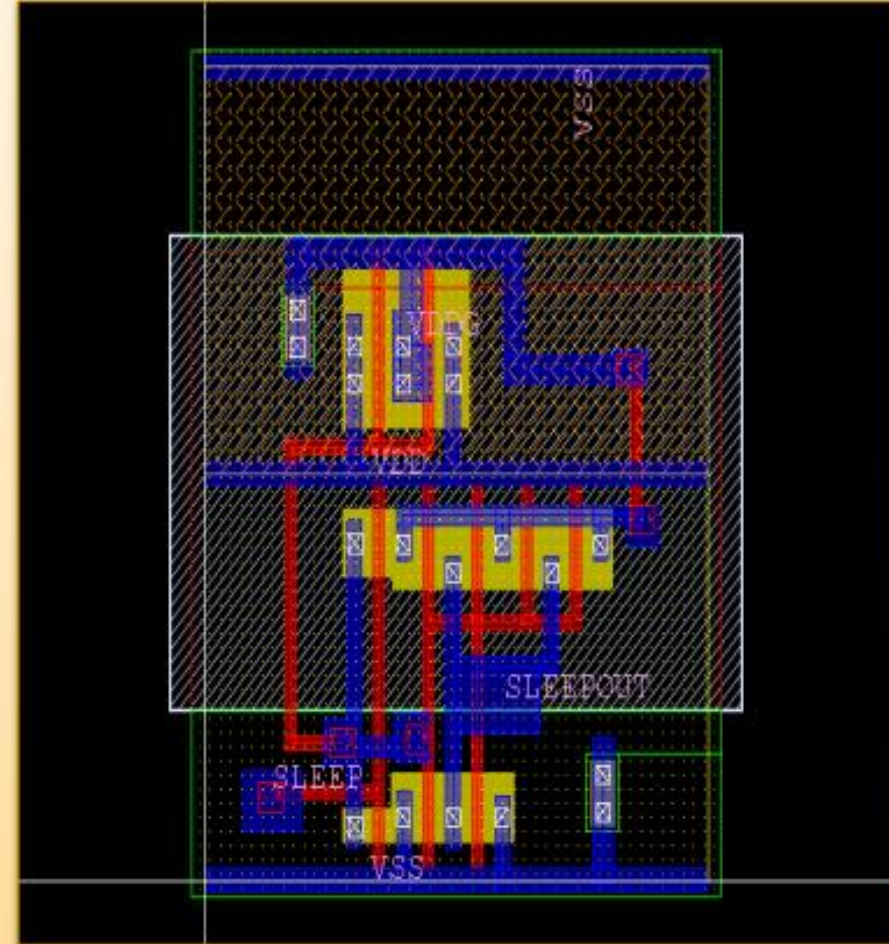


Source: Keating

# Header Cells Layout



a. Header Cell



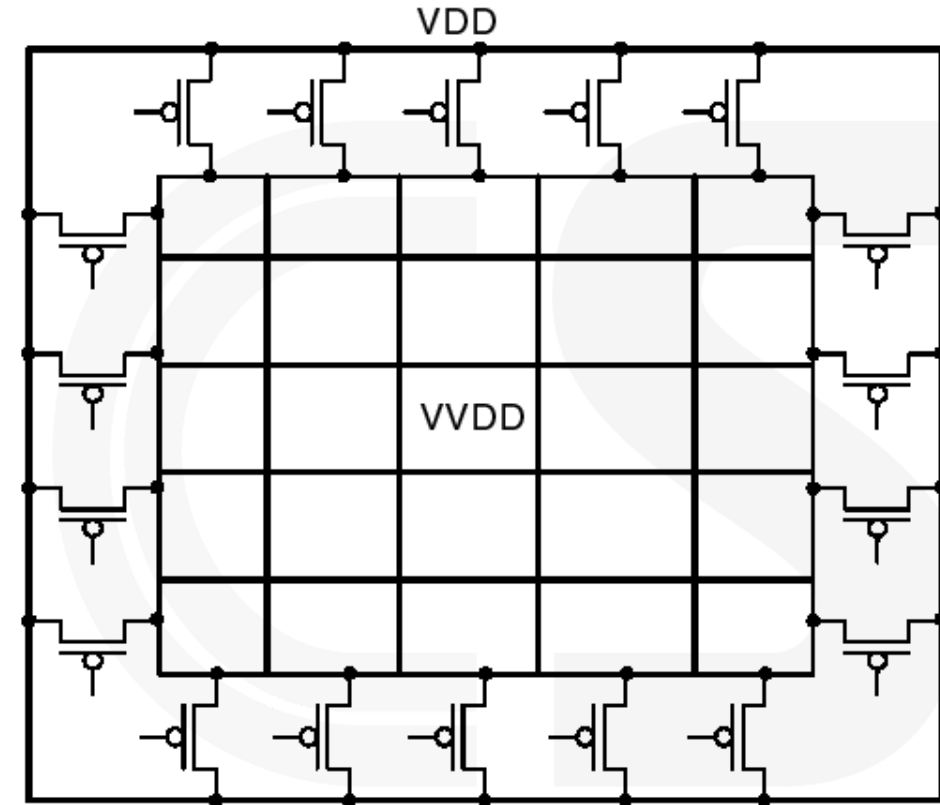
b. Header Cell (with SLEEPOUT output)

Source: Synopsys

© Adam Toman, 2020

# Ring Style Implementation

- A ring of VDD surrounds the power gated block.
- **Advantages:**
  - Simple power plan
  - Little negative impact on PNR
  - Always-on cells can be placed around the power domain areas.
- **Disadvantages:**
  - Large IR Drop in big power domains
  - Does not support retention registers
  - High extra area cost compared to a grid approach.
- **Note that a ring style is the only option to power gate an existing hard block.**

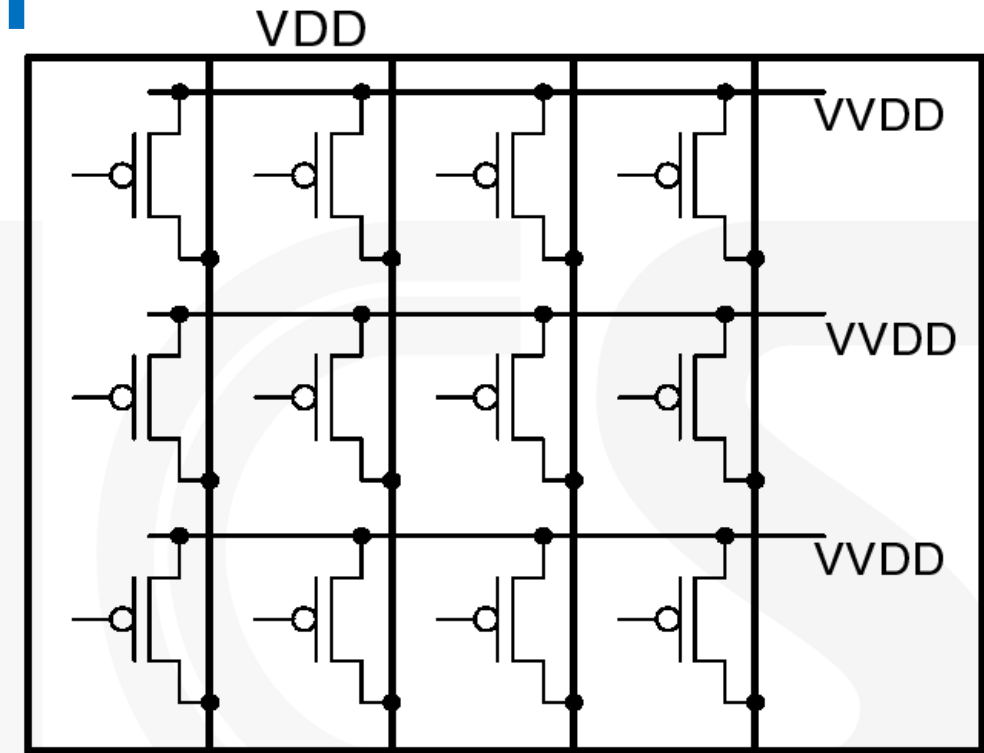


Source: Keating



# Grid Style Implementation

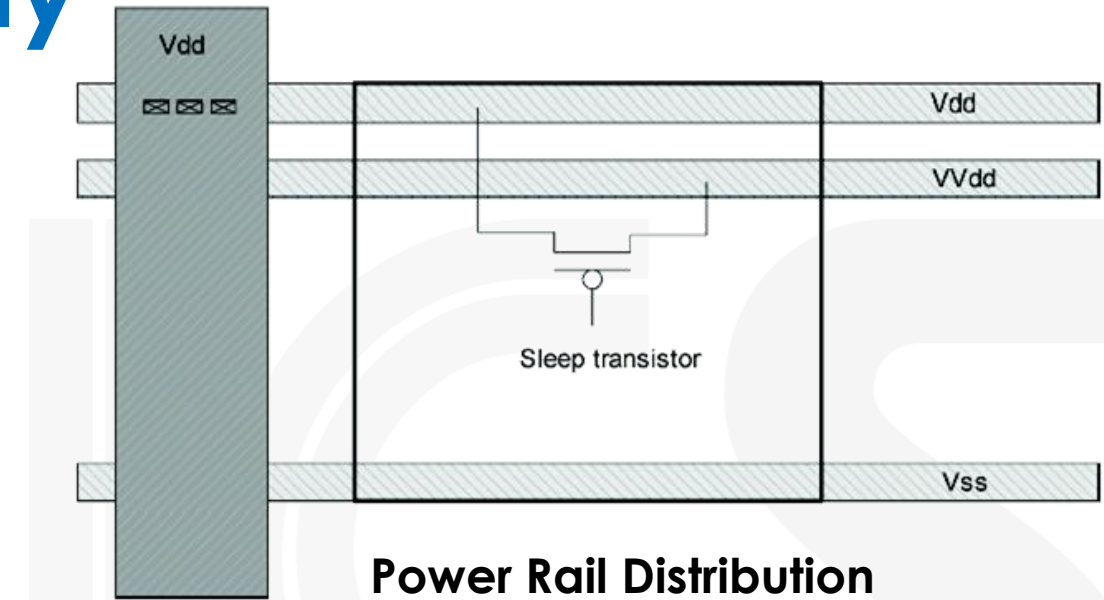
- The sleep transistors are distributed throughout the power gated region.
- **Advantages:**
  - Virtual supply can be routed in low metal layers
  - Fewer sleep transistors than ring-style to achieve the same IR drop target.
  - Retention registers and always-on buffers can connect to the always on supply anywhere within the domain
  - Better trickle charge distribution for management of in-rush current.
  - Lower area overhead (due to <100% placement utilization).
- **Drawbacks**
  - Impact on standard cell routing and physical synthesis.
  - Much more complex power routing



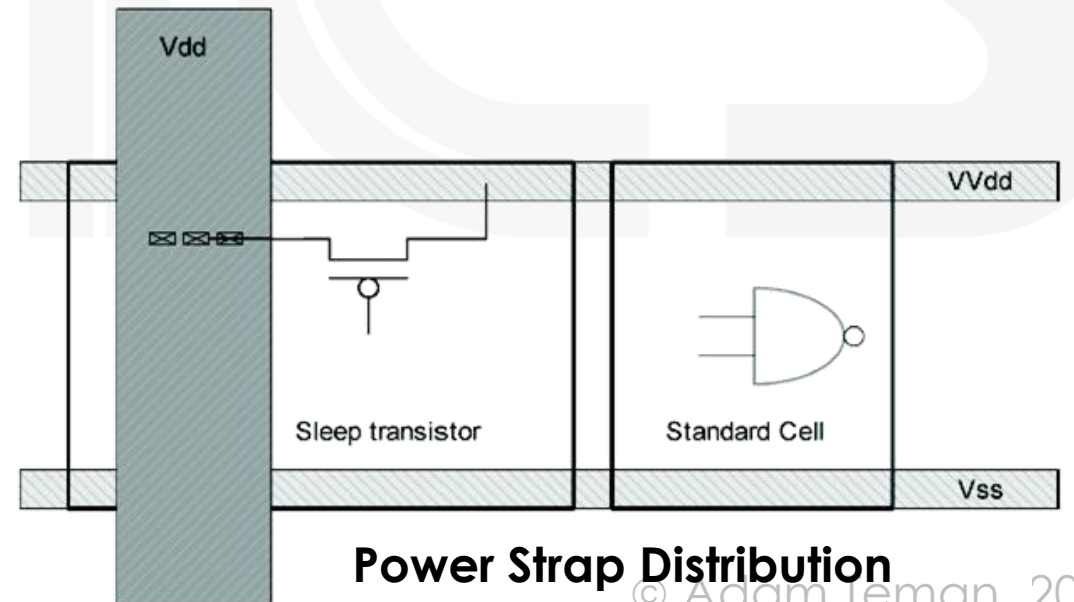
Source: Keating

# Rail vs. Strap VDD Supply

- **Two options for supplying VDD:**
  - Parallel Rail VDD Distribution
  - Power Strap VDD Distribution
- **Parallel Rail VDD Distribution**
  - Easy access to VDD and VVDD
  - High overhead (at least 1 track per row)
- **Power Strap VDD Distribution**
  - No need for specialized SC library
  - No need for extra row
  - Placement of special cells under the strap or connected through power route
  - Potentially worse IR drop



Source: Keating



© Adam Ieman, 2020

# Limiting In-Rush Current

- **In-Rush Current**

- When a block is switched on, a strong “in-rush” current occurs
- Can cause supply voltage spikes, possibly disrupting always on and retention registers

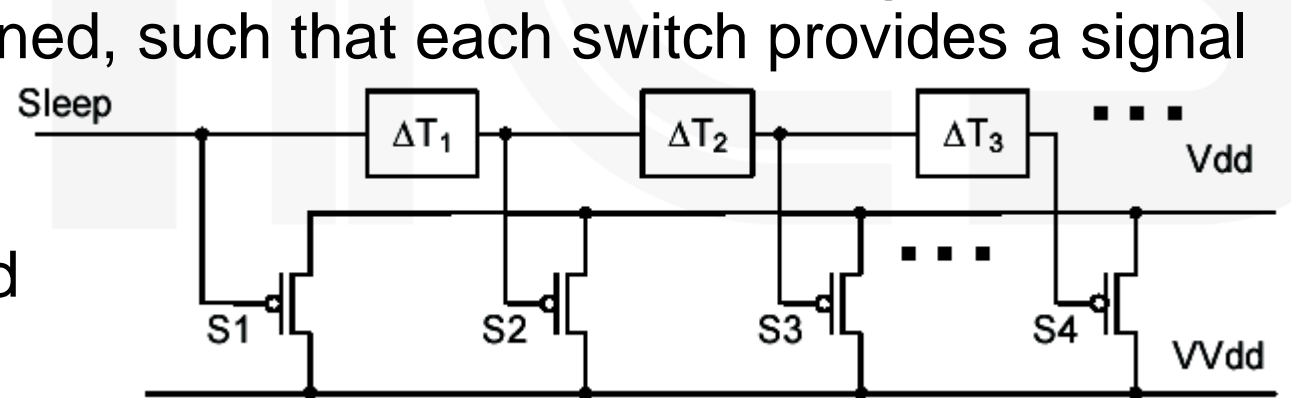
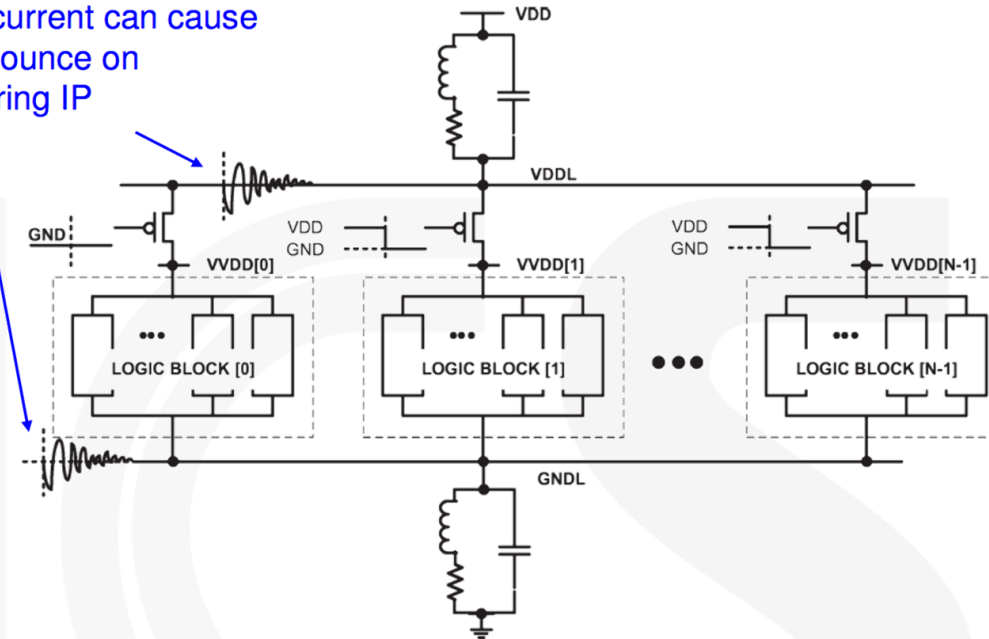
- **Daisy Chaining**

- Control signals can be daisy chained, such that each switch provides a signal to enable the next switch.
- This requires a substantial delay, so a ready signal is usually added

- **Trickle Switches**

- First turn on a set of weak switches, which initiate power up with limited in-rush
- Then turn on the main set of switches

In-rush current can cause supply bounce on neighboring IP





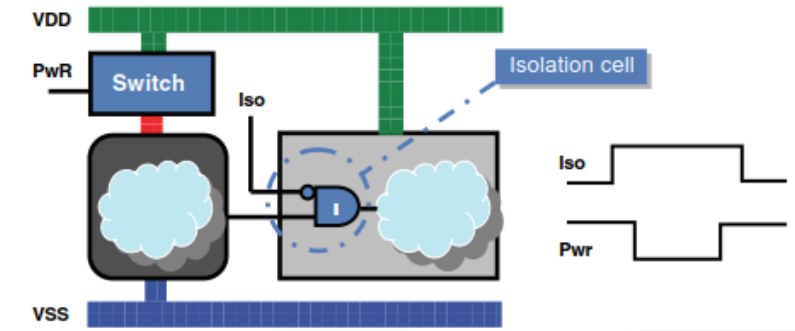
# Signal Isolation

- **The problem:**

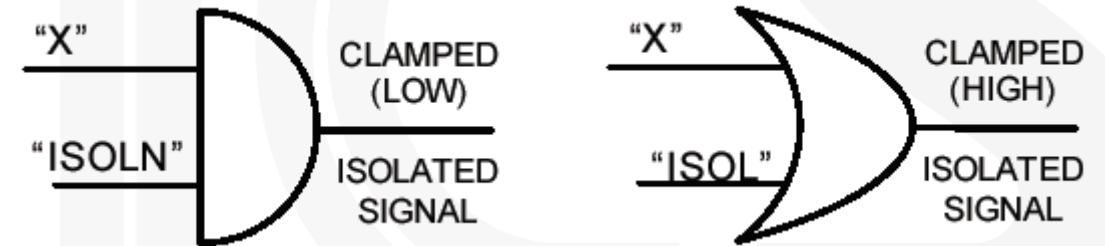
- When a block is powered down, the outputs are floating.
- If a floating output drives an always-on region, crowbar current can occur.
- Note that inputs are not a problem, if they are driven by always-on signals.

- **The solution:**

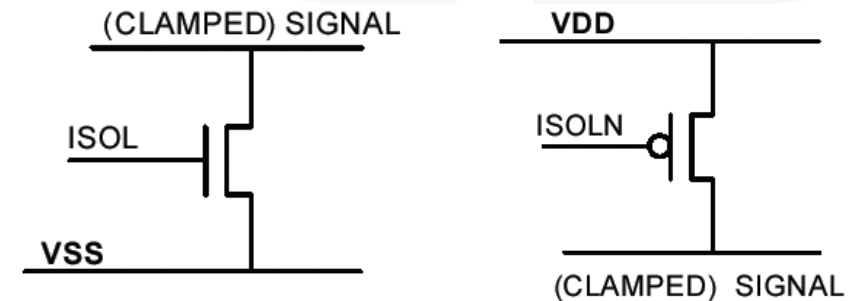
- Use an isolation clamp
- Usually clamp to an inactive state:
  - Active high logic: Clamp to a '0' using an AND gate.
  - Active low logic: Clamp to a '1' using an OR gate.
- To limit added delay, pull-up/pull-down clamps can be used
  - This approach is much more problematic and so is seldom used.



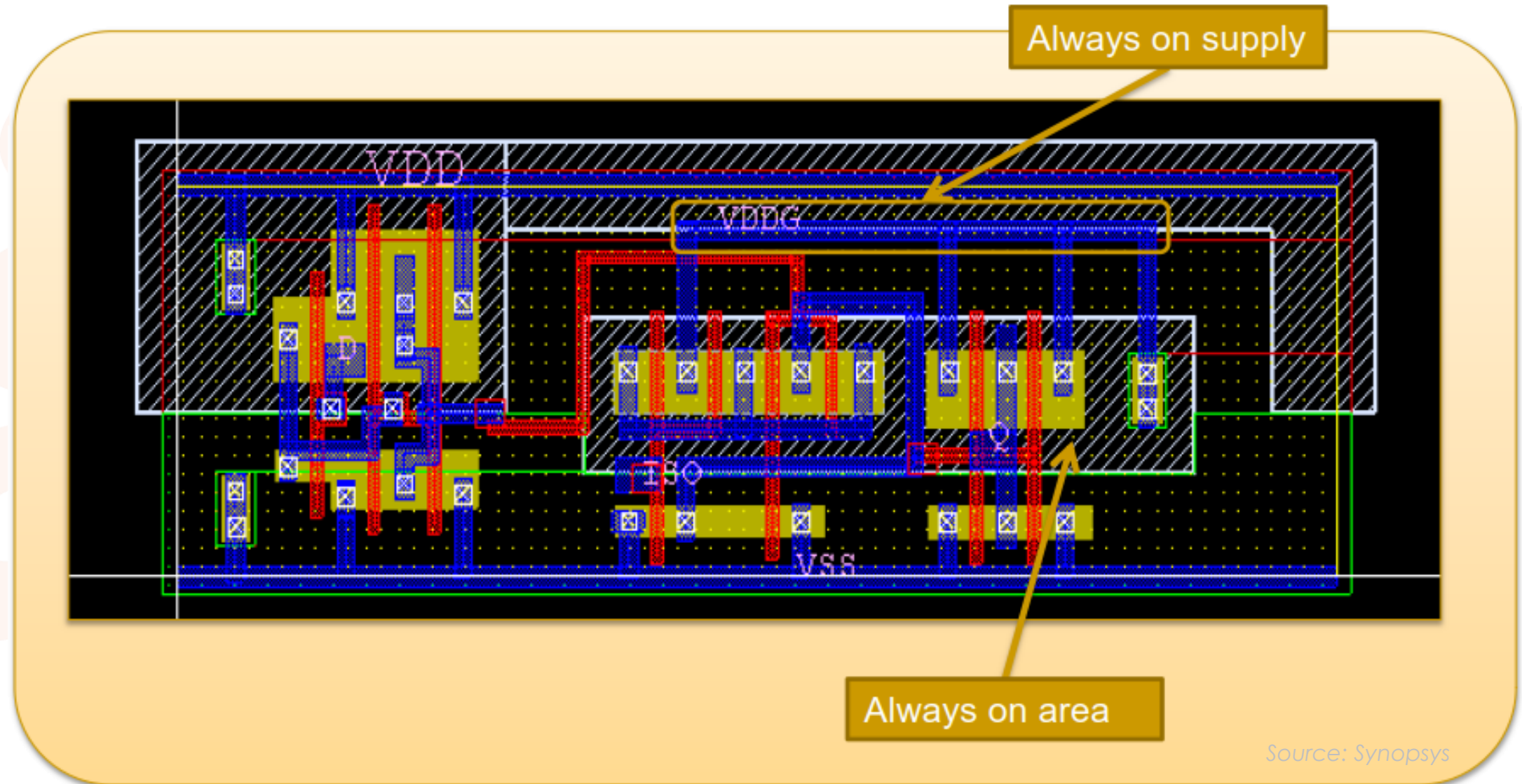
Source: power forward



Source: Keating

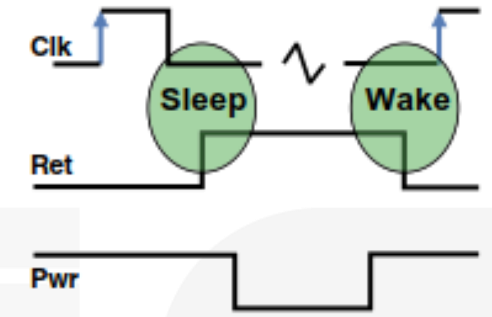
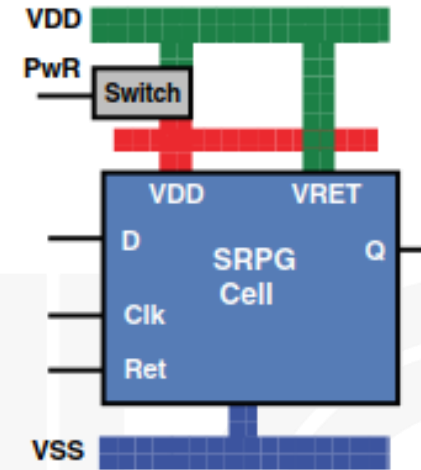


# Always-on Isolation Cell Layout



# State Retention

- When powering down a block, all state information is lost. To resume its operation:
  - Restore the state from an external source.
  - Build up its state from the reset condition.
- **Instead, use a retention strategy for quick state restoration**
  - This is recommended for a peripheral or cached processor with significant residual state.
- **Methods for Saving and Restoring the Internal States:**
  - A software approach based on reading and writing registers
  - A scan-based approach based on using scan chains to store state off chip
  - A register-based approach that uses retention registers



Source: power forward

# Retention Approaches

- **Software-based approach**

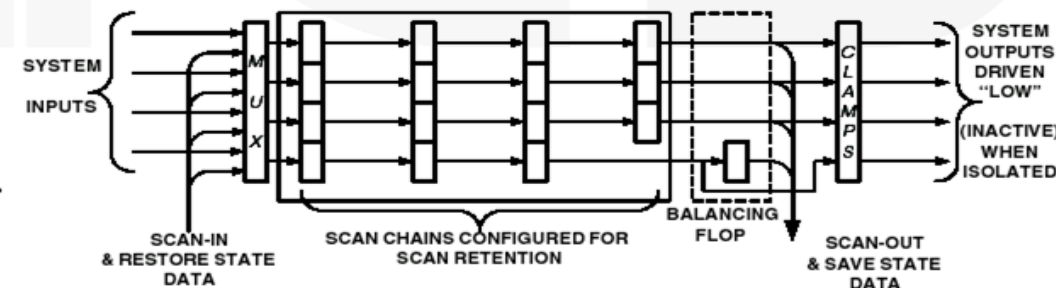
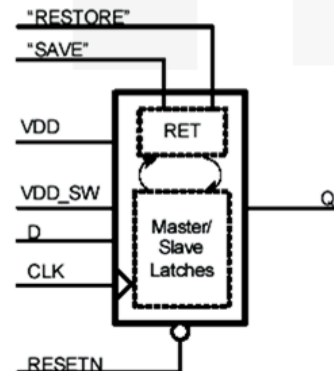
- An always-on processor reads and stores the registers of the power-gated block during the power shutdown sequence
- This suffers from slow power-down/power-up sequences that are non-deterministic due to bus conflicts, as well as very tightly-coupled software

- **Scan Chain based approach**

- Separate scan chains according to power domains and scan out state to external memory for retention.

- **Retention Register approach**

- A retention register contains an always-on “shadow” register
- Typically 20%-50% overhead



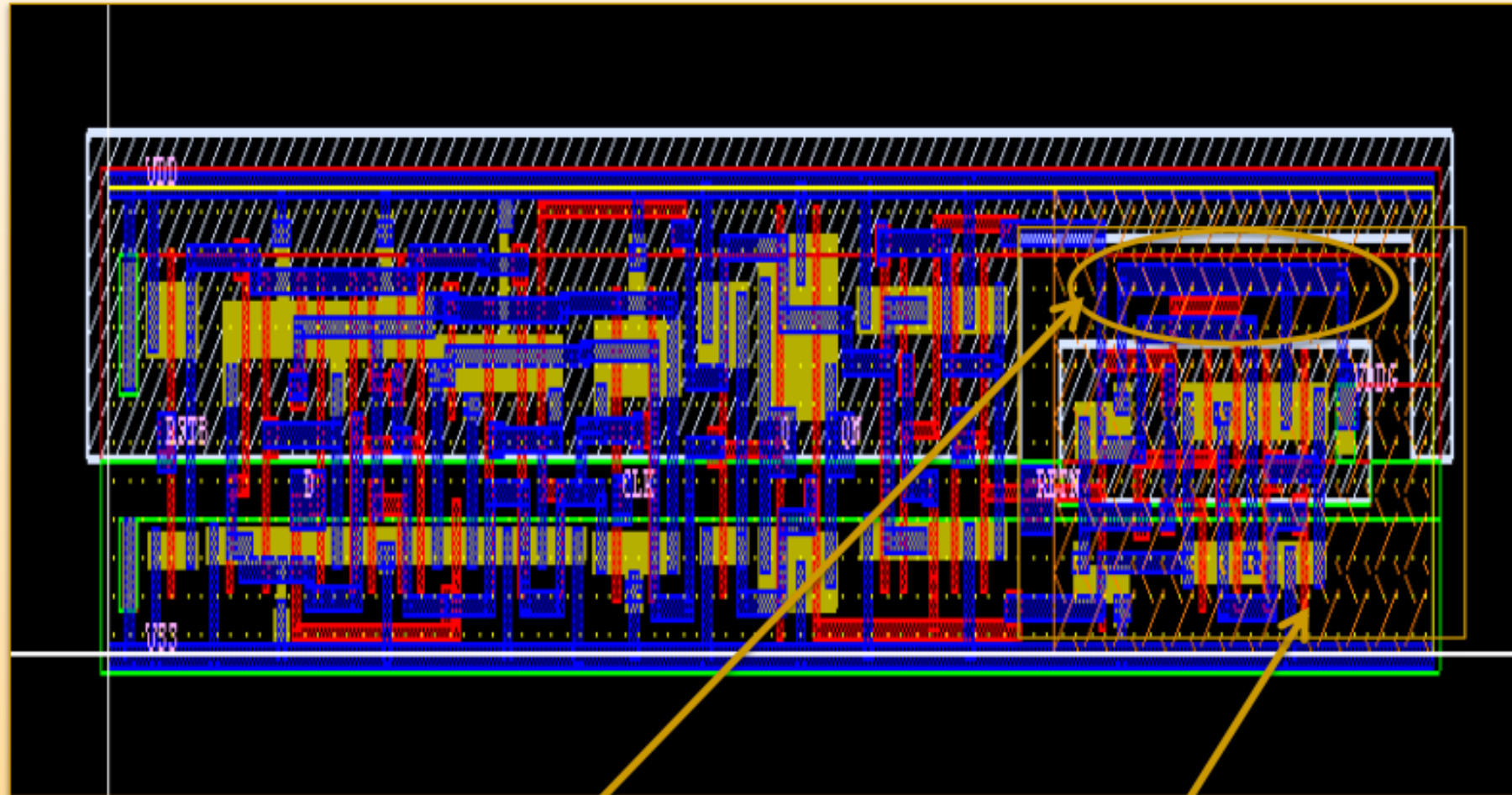
Source: Keating

Source: Keating

© Adam Teman, 2020



# Retention Register Layout



Always-on Power pin

Always on area, with high-Vth

Source: Synopsys

© Adam Teman, 2020

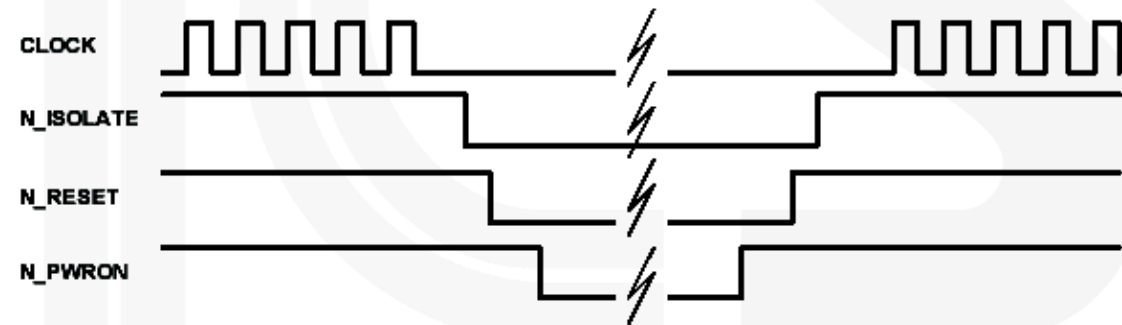
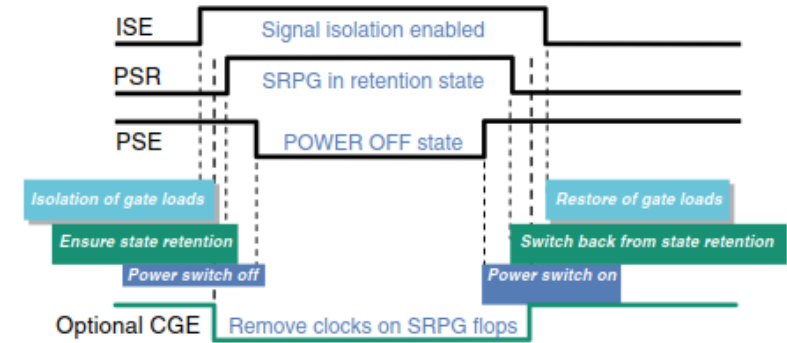
# Power Control Sequencing

- To power gate a region without retention:

- Flush through any bus or external operations in progress
- Stop the clocks in the appropriate phase to minimize leakage
- Assert the isolation control signal to park all outputs in a safe condition
- Assert reset to the block, so that it powers up in the reset condition
- Assert the power gating control signal

- To restore power:

- De-assert the power gating control signal
- Optionally sequence multiple control signals for phased power-up depending on the current in-rush management approach and technology
- De-assert reset to ensure clean initialization following the gated power-up
- De-assert the isolation control signal to restore all outputs
- Restart the clocks, without glitches and without violating



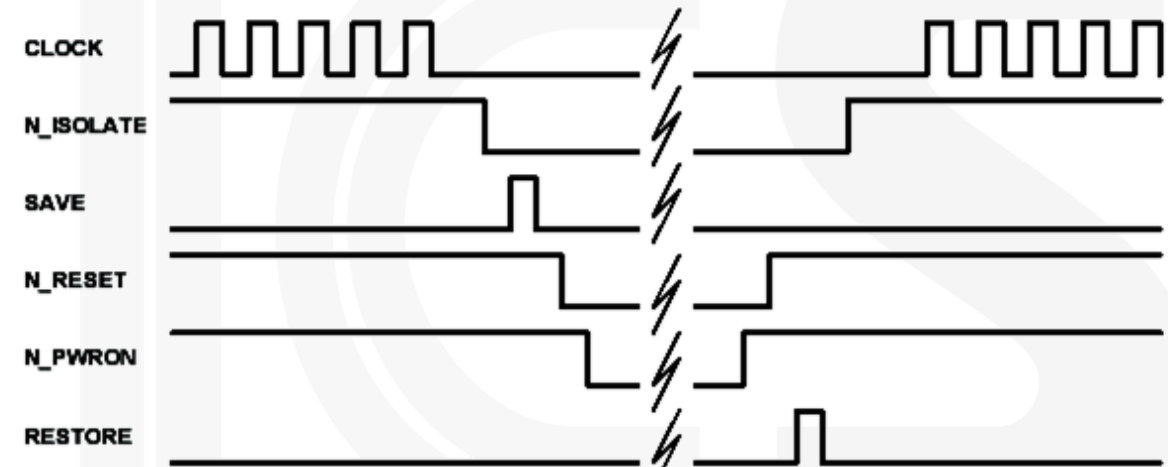
# Power Control Sequencing

- **To power gate a region with retention:**

- Flush through any bus or external operations in progress
- Stop the clocks
- Assert the isolation control signal
- Assert the state retention save condition
- Assert reset to the non-retained registers, so that they power up in the reset condition
- Assert the power gating control signal

- **To restore power and retained state:**

- De-assert the power gating control signal
- Optionally sequence multiple control signals for phased power-up
- De-assert reset to ensure clean initialization following the gated power-up
- Assert the state retention restore condition
- De-assert the isolation control signal
- Restart the clocks without glitches



Source: Keating

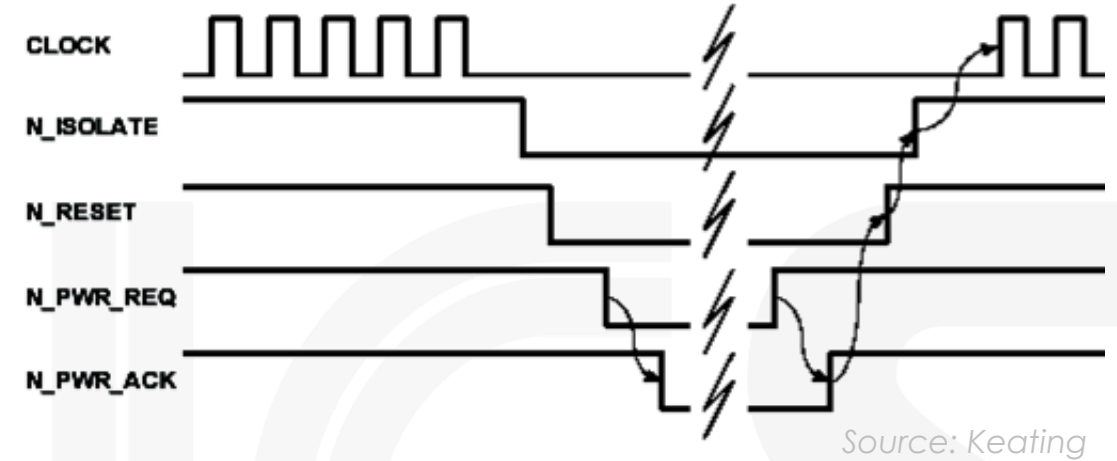
# Handshake Protocols

- **Two approaches to delayed wake-up:**

- Fixed-delay (counter based)
- Using a request-acknowledge handshake.

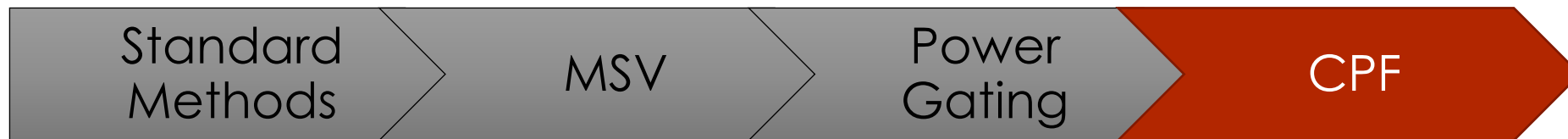
- **Handshake protocol**

- The power controller issues a N\_PWR\_REQ to turn the power switching fabric off.
- It is the responsibility of the switching fabric to return N\_PWR\_ACK when power is completely switched off.
- On power up, the controller de-asserts N\_PWR\_REQ to turn the switching fabric on.
- When the fabric is completely on and it is safe to proceed, the switching fabric deasserts N\_PWR\_ACK.
- When the controller sees the acknowledge, it proceeds to assert restore and continue through the power up sequence.



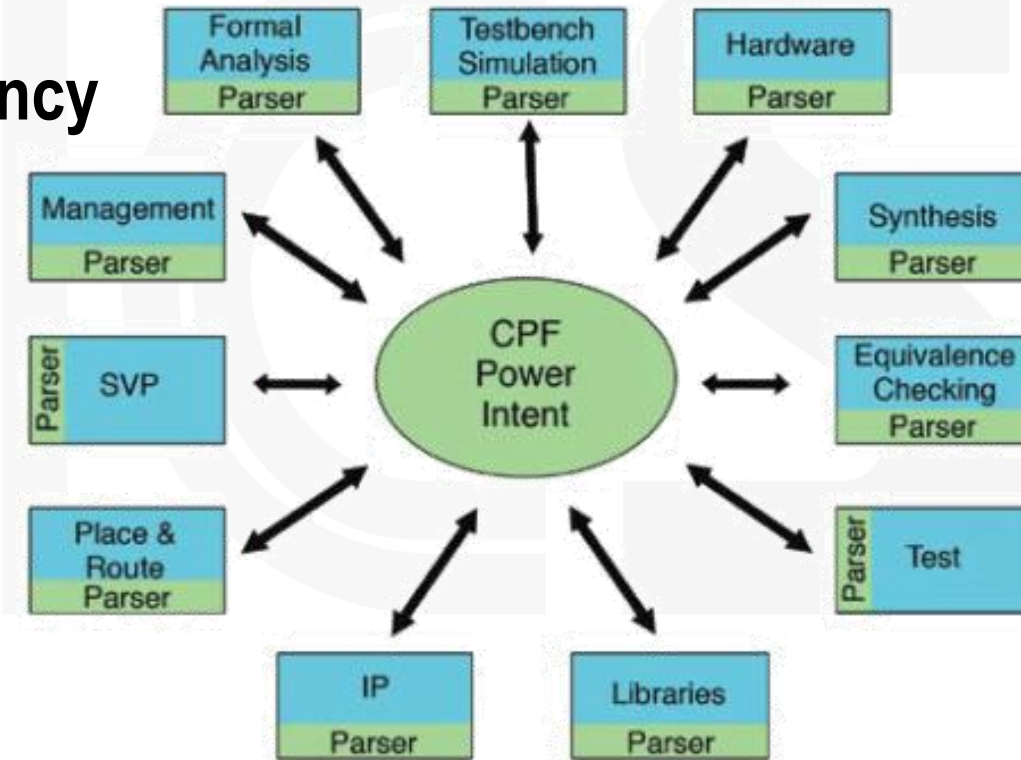
Source: Keating





# The Need for a Common Power Format

- Low-power design flows need to specify the desired power architecture to be used at each major step and for each task.
- Old flows had no way of guaranteeing consistency
- **Benefits of CPF:**
  - Improved quality of silicon (QoS)
  - Higher productivity and faster time to market
  - Reduced risk
- **Two formats:**
  - UPF
  - CPF
- We will be showing a CPF example in the following slides



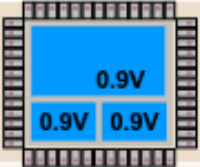
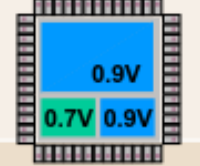
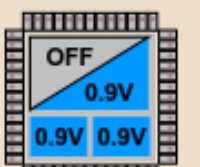
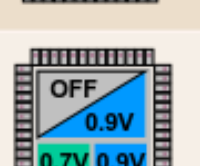
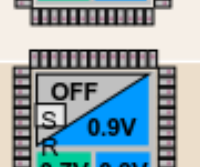
# Special Cells for Low Power Techniques

`switch_cell_type : coarse_grain;`

`is_level_shifter : true;`

`is_isolation_cell : true;`

`retention_cell : cell_type;`

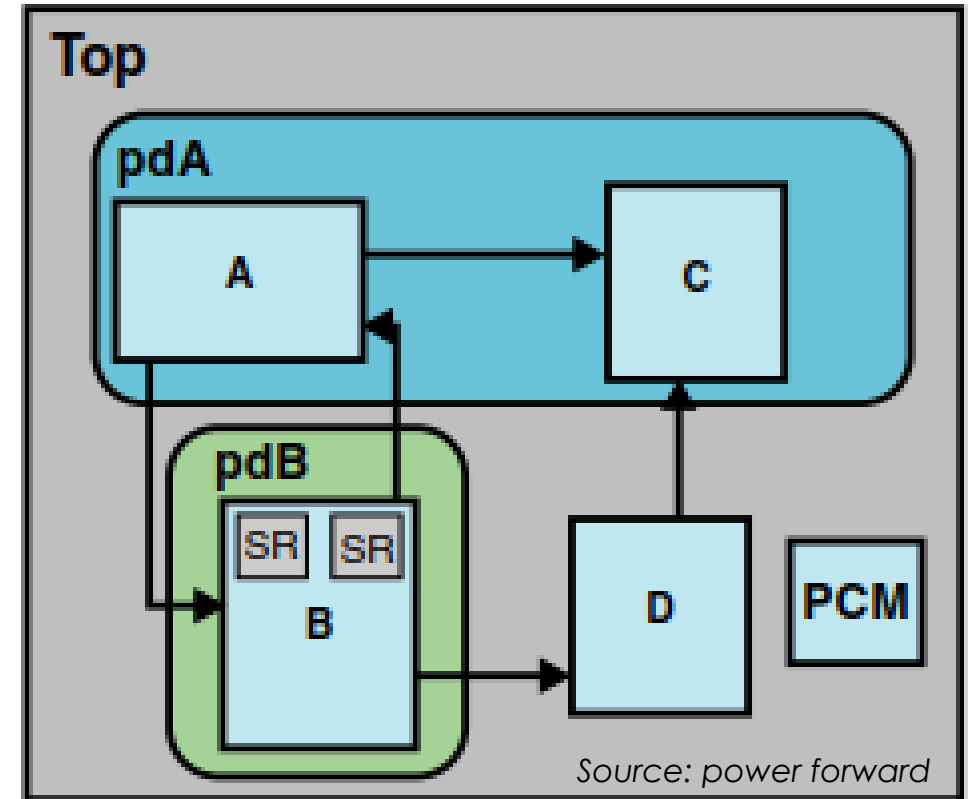
		Power Gates (MTCMOS)	Isolation Cells	Level Shifters	Retention Registers	Always On Logic
	Multiple Power Domains Single Voltage					
	Multiple Voltage (MV) Domains			+		
	Power Gating (shut down) Single Voltage No State Retention	+	+			+
	MV Domains Power Gating No State Retention	+	+	+		+
	MV Domains Power Gating State Retention	+	+	+	+	+

# CPF through example

- Define power domains

- Define power domains
- Define power nets for domains
- Specify power down signals (from PCM unit)

```
# Define the top domain
set_design TOP
# Define the default domain
create_power_domain -name pdTop -default
update_power_domain -name pdTop -internal_power_net VDD
# Define PDA - PSO when pso is low
create_power_domain -name pdA -instances {uA uC} \
    -shutoff_condition {!uPCM/pso[0]}
update_power_domain -name pdA -internal_power_net VDDa
# Define PDB - PSO when pso is low
create_power_domain -name pdB -instances {uB} \
    -shutoff_condition {!uPCM/pso[1]}
update_power_domain -name pdB -internal_power_net VDDb
```



# Define operating voltages

- First define “nominal conditions”
- Then define libraries for each voltage
- And connect the condition to the libraries
  - Also need to create operating corners...

```
# Nominal operating conditions
```

```
create_nominal_condition -name high -voltage 1.2
update_nominal_condition -name high -library_set lib_high
create_nominal_condition -name medium -voltage 1.0
update_nominal_condition -name medium -library_set lib_med
create_nominal_condition -name low -voltage 0.8
update_nominal_condition -name low -library_set lib_low
create_nominal_condition -name off -voltage 0
```

```
# Define libraries
```

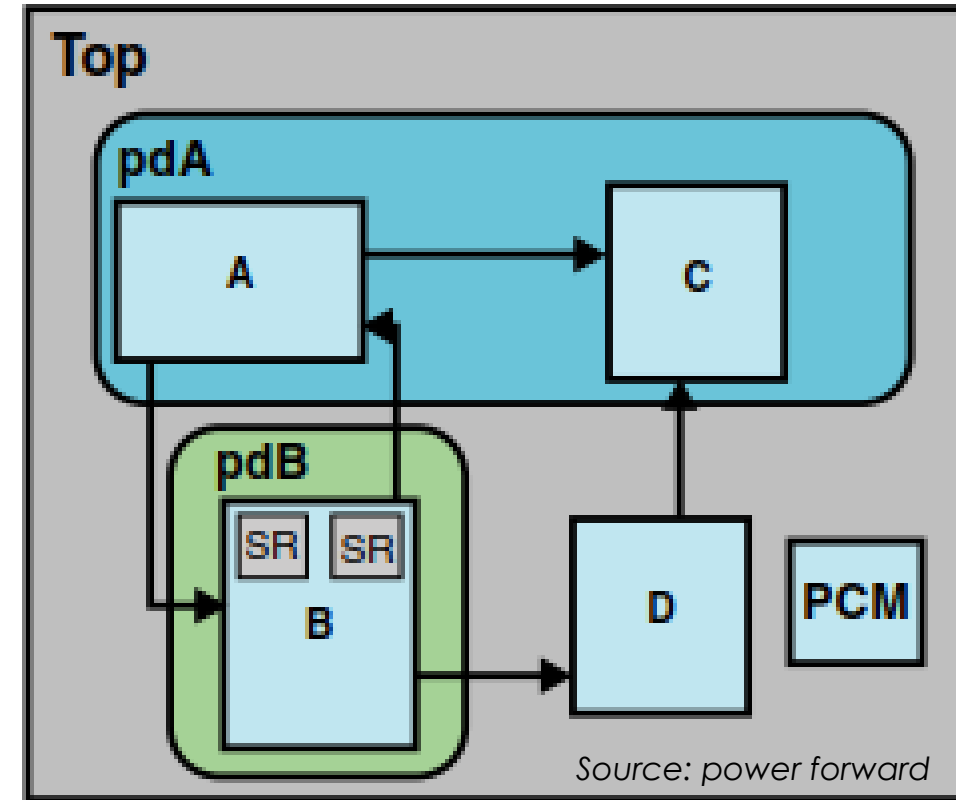
```
define_library_set -name lib_high -libraries Lib1
define_library_set -name lib2_med -libraries {Lib2 Lib3}
define_library_set -name lib3_low -libraries Lib4
```

```
create_operating_corner -name corner1 \
    -voltage 0.80 -process 1 \
    -temperature 125 -library_set lib_low
create_operating_corner -name corner2 \
    -voltage 1.0 -process 1 \
    -temperature 125 -library_set lib_med
create_operating_corner -name corner3 \
    -voltage 1.2 -process 1 \
    -temperature 125 -library_set lib_high
```

# Define Power Modes

- Finally create “power modes”
  - In PM3, pdB is powered down
- Define constraints for each mode
- Connect the operating corners

```
create_power_mode -name PM1 -default \  
  -domain_conditions {pdTop@high pdA@medium pdB@medium}  
update_power_mode -name PM1 -sdc_files PM1.sdc  
create_power_mode -name PM2 \  
  -domain_conditions {pdTop@high pdA@low pdB@low}  
update_power_mode -name PM2 -sdc_files PM2.sdc  
# Mode where pdB is off  
create_power_mode -name PM3 \  
  -domain_conditions {pdTop@high pdA@low pdB@off}  
update_power_mode -name PM3 -sdc_files PM3.sdc  
create_analysis_view -name PM1_view -mode PM1 \  
  -domain_corners {pdTop@corner1 pdA@corner2 pdB@corner3}
```

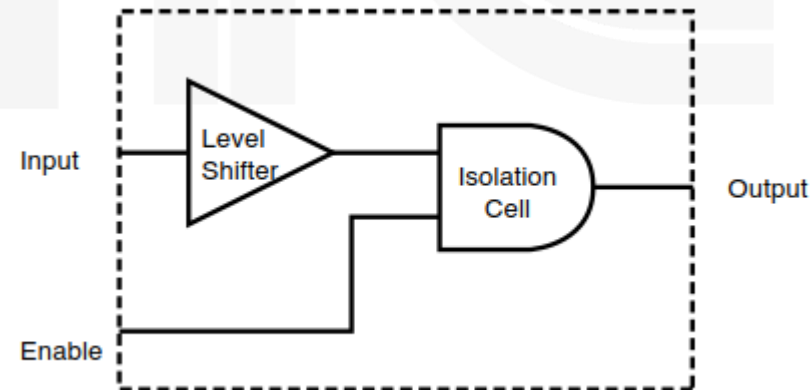


# Defining Level Shifters

- In order to automate the insertion of level shifters during synthesis or PNR, they need to be defined

```
define_level_shifter_cell \  
  -cells LVLHVT -valid_location from \  
  -input_voltage_range 0.8 \  
  -output_voltage_range 1.2 -ground VSS \  
  -input_power_pin VDD \  
  -output_power_pin VDDH
```

- Level shifters can also be combined with isolation cells



Source: power forward



# Defining Level Shifters

- But you also need to define rules
  - First define the connectivity
  - Then define the cells and location

```
# Define Level-Shifters in the "to" domain
```

```
create_level_shifter_rule -name lsr1 \  
  -to {pdB} -from {pdA}
```

```
update_level_shifter_rules -names lsr1 \  
  -cells LS32 -location to
```

```
create_level_shifter_rule -name lsr2 \  
  -to {pdA} -from {pdB}
```

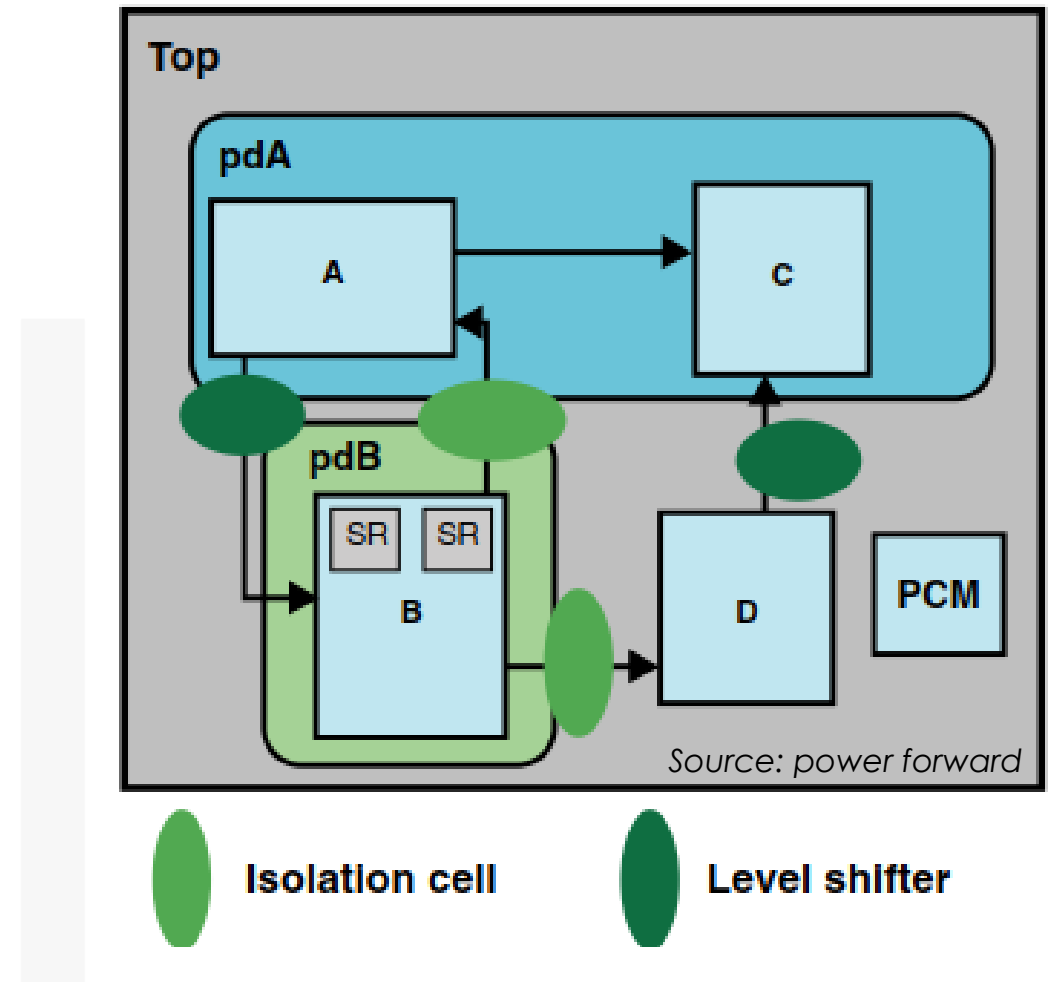
```
update_level_shifter_rules -names lsr2 \  
  -cells LS23 -location to
```

```
create_level_shifter_rule -name lsr3 - to {pdTop} -from {pdB}
```

```
update_level_shifter_rules -names lsr3 -cells LS13 -location to
```

```
create_level_shifter_rule -name lsr3 -to {pdA} -from {pdTop}
```

```
update_level_shifter_rules -names lsr1 -cells LS21 -location to
```



# Power Switch and Isolation Cells

- For power gating, power switch cell definitions are required:

```
define_power_switch_cell -cells HDRHVT \  
-stage_1_enable SLPIN -stage_1_output SLPOUT \  
-power VDDH -power_switchable VDDI  
create_power_switch_rule -name PSW_RULE -domain ALUP
```

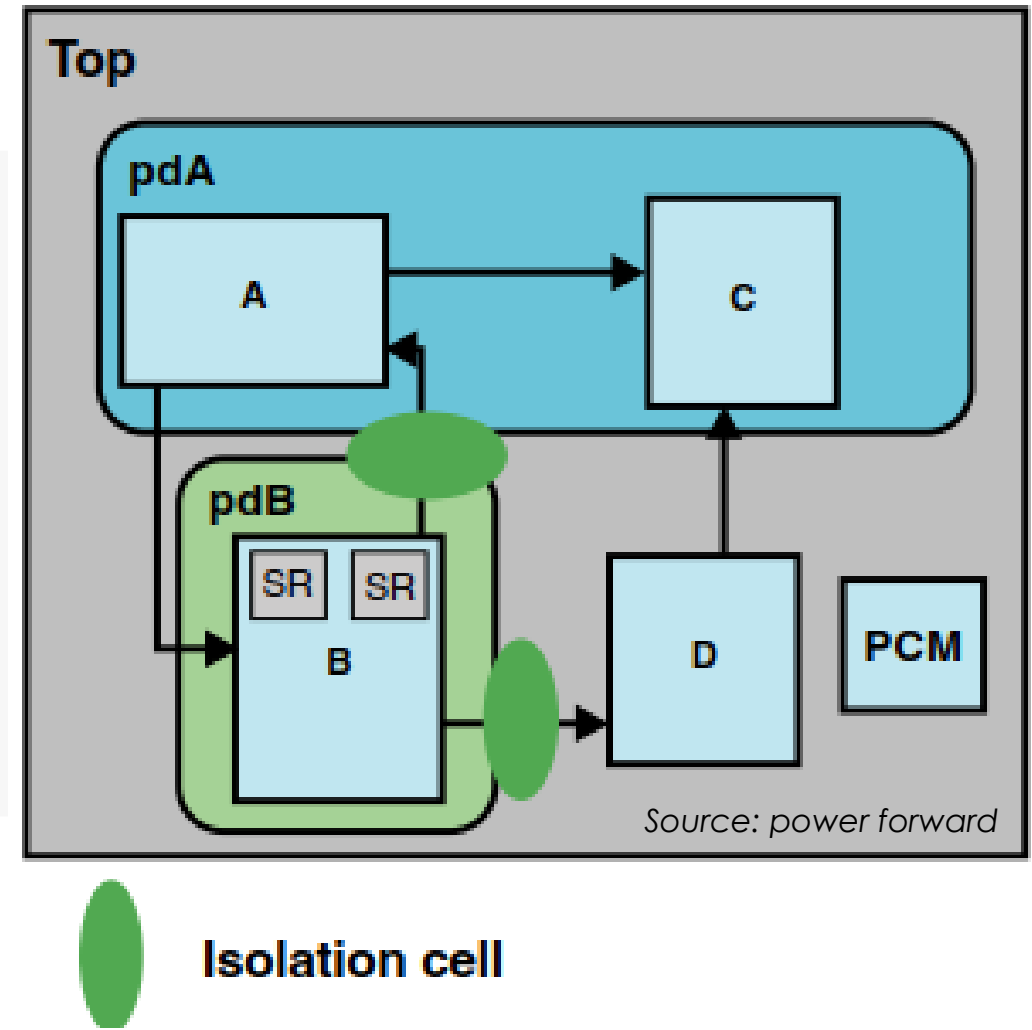
- As well as Isolation Cell definitions:

```
define_isolation_cell -cells ISOHVT \  
-enable NSLEEP -power VDD -ground VSS
```

# Adding in Isolation

- Define rules for adding isolation cells
  - Specify isolation signal (from PCM unit)
  - Isolation value can be high, low or hold
  - Specify the cells to use

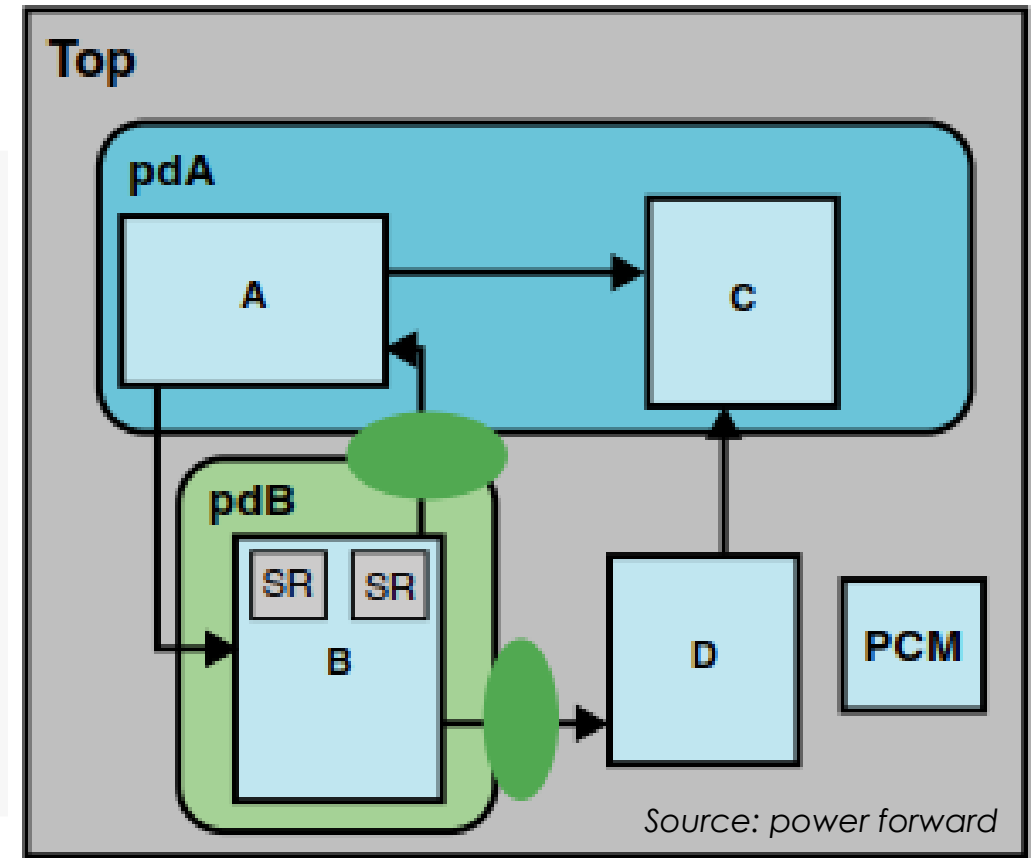
```
## All outputs of Power-Domain pdB
# isolated high on rising edge of "iso"
create_isolation_rule \
  -name ir1 \
  -from pdB \
  -pins {uB/en1 uB/en2} \
  -isolation_condition {uPCM/iso} \
  -isolation_output high
update_isolation_rules -names ir1 -cells ISOLS2 \
  -combine_level_shifting -location to
```



# Defining State Retention

- Define state retention requirements
  - Registers to be saved
  - Signal to trigger restoration
- Also need to define the cells to use

```
## Define State-Retention (SRPG)
# State stored on falling edge of restore[0]
# and restored on rising-edge
create_state_retention_rule \
  -name sr1 \
  -instances {uB/reg1 uB/reg2} \
  -restore_edge {uPCM/restore[0]}
# -save is by default !restore_edge
update_state_retention_rule -names SRPG1 \
  -library_set lib_med -cell_type DRFF
```



Isolation cell

# Main References

- **“Low Power Methodology Manual for SoC Design”**, M. Keating, D. Flynn, et al.
- **“A Practical Guide to Low-Power Design”**, The Power Forward Initiative