Showcasing Efficient Low-Power Techniques Using RISC-V CPU Cores with DC-NXT And ICC2 Using GF's 22FDX-PLUS technology

Farid Labib, Stefan Block, Juergen Dirks, Fulvio Pugliese,
Chaithra Kyathanahally Ramesh, Ulrich Hensel,
Siva Kumar Anala, Falk Tischer, Toni Laaksonen, Joerg Winkler

GlobalFoundries

Munich, Dresden, Bangalore

www.gf.com

ABSTRACT

With the growing need for low power circuits in small battery-powered devices, it is crucial to achieve implementations which are energy efficient as well as area optimized. The present paper demonstrates the capabilities of GF's 22FDX-PLUS Ultra High-Density standard cell low power kit from Synopsys, in achieving power and area optimized implementations for a configuration of a 4 Core compute cluster (4 instances of RISC-V CPU cores) using DC-NXT and ICC2. Special focus was put on the insertion of Retention-Flip-Flops, Power-Switches and Always-On-Buffers, in the context of an area-efficient secondary power-mesh. Additionally, potential challenges are addressed, and solutions provided. We will also briefly touch on the value proposition of this library.

Table of Contents

Table of Figures

Table of Tables

Abbreviations

ABB	Adaptive Body Bias
EMIR	Electromigration and IR-Drop
LEQ	Logic Equivalence Checking
RISC	Reduced Instruction Set Core
STA	Static Timing Analysis
TCM	Tightly Coupled Memory
UPF	Unified Power Format (IEEE-1801)

Table 1: Abbreviations

Tool and rm-script versions

Tool	Version	rm-scripts version
DC-NXT	2022.12-SP7	U-2022.12-SP4
ICC2	2022.12-SP6-T-20231012	U-2022.12-SP6
Redhawk	2022R2.3	-

Table 2: Tool and rm-script versions

1. Introduction

The purpose of this paper is to share best practices from discoveries made during implementation of a RISC-V design for specific low power goals using area optimized standard cell libraries. Please, note, the material consists of general descriptions of techniques which might be known to experts, while pointing out specific difficulties and work arounds for them, which makes the present paper a comprehensive summary of guidelines for various power implementation topics.

A key motivation for using an area optimized library is overall cost savings caused by design area reduction. As a side effect a reduced design area can also help with power savings. On the other hand, dense libraries may require special handling by synthesis and place&route tools. Therefore, implementation tools need to be properly guided, otherwise they may require overhead which in turn reduces the originally intended improvements with using the area optimized standard cell libraries.

As a result of the implementation trials the present paper focuses on two main aspects:

1. General area optimization (standard cell placement, routing directions, macro placement)
2. Implementing specific power saving techniques

With the respective tool guidance, overall utilization and area could be optimized, followed then by implementation of low power techniques.

The present paper does not discuss topics of power intent verification, STA, and in-rush current analysis.

2. The Design

The design example is based on the open-source “PULP KRAKEN” design by ETH Zurich (Figure 1). The present paper uses the “Compute Cluster” module “pulp_cluster”, which implements 4 RISC-V CPU cores together with a shared bank of Tightly Coupled Memories (TCM) and related control logic. To achieve the first level of power reduction the design is going to be implemented for a low voltage of 0.65 V nominal and therefore operating at a speed of 200 MHz. To achieve the performance goal adaptive forward body bias is going to be applied to standard cells and SRAMs.

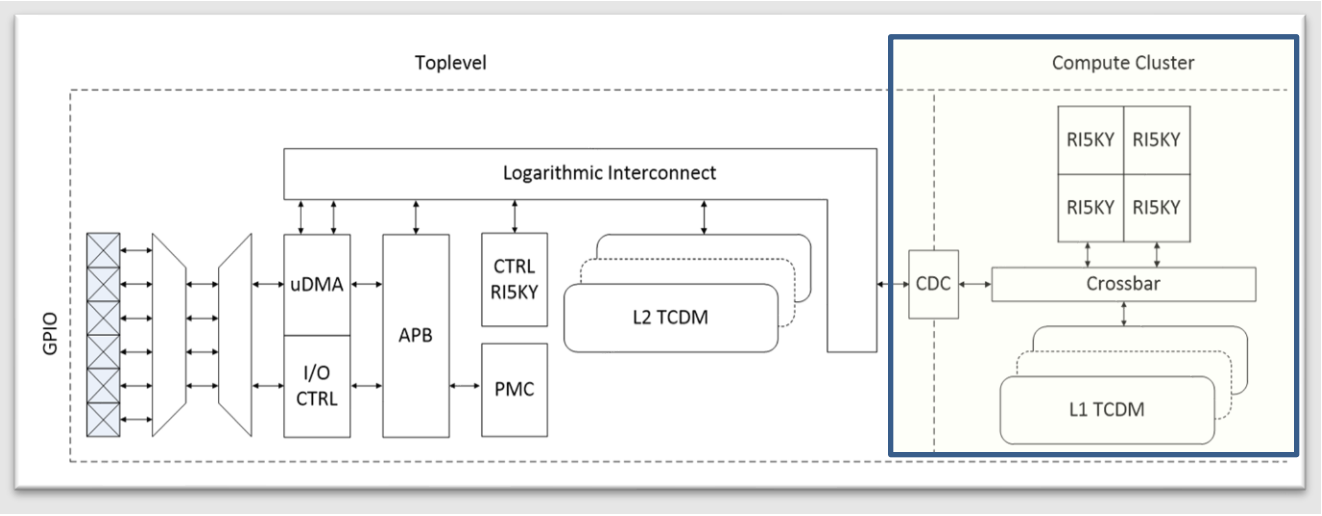


Figure 1: Block diagram of “Kraken” toplevel and “Compute Cluster” (module pulp_cluster)

3. The Technology and IP Selection

The target technology for the present paper is GF's 22FDX-Plus, a 22nm fully depleted SOI technology (FDSOI). However, the techniques illustrated herein are not limited to this technology. A cost saving metal stack with 6 routing layers plus one thick layer is used.

As a next step towards a low power implementation the selected standard cell library uses an ultra-dense and ultra-low power cell architecture, leveraging multi-height cells, such as multi-bit flip-flops.

From the power management kit of this library, power switches, always-on buffers, retention flip-flops and isolation cells are pulled. In particular, usage of retention flip-flops is highlighted, as it appears, this is not widely used. Retention flip-flop can be key to overall energy saving.

For the L1 cache, dual rail memories (separate power for bit-array and periphery), with deep shut-down mode are used.

Adaptive Forward Body Bias (ABB) is enabled, to allow tightening of the slow corners, to further reduce overall power. This requires some additional, but minor considerations during the power mesh planning.

4. Implementation

The focus of the implementation is split into two parts:

1. Design Area Reduction
2. Low Power Techniques

Initial experiments will be done without power switches, to concentrate on overall area optimization. In a second step, the design will be partitioned to enable switchable power domains, including the necessary supporting cells and mesh structure. Note, Retention Flip Flops and a certain amount of secondary power are already part of the first experiments to account for the required overhead.

4.1 Synthesis with DC-NXT

For synthesis, Synopsys' recommended flow is used. The UPF (version 2.1) is fully coded, and follows the desired power intent, as shown in Figure 5 further down. The so-called "prime flow" is used, across the entire implementation, meaning DC-NXT will receive the input UPF, then writes out all additionally added multi-voltage cells, such as isolation, always-on buffers, retention flip-flops. ICC2 then reads in the output UPF from DC-NXT. The full UPF code can be found in the Appendix.

Remark: Our initial trials utilized the "golden UPF" flow, where one golden UPF is used by all tools along the flow, and only incremental updates are written out by one tool and passed over to the next tool. During review sessions with Synopsys, the proposal was to go with the prime flow, as golden flow is not widely used.

A few aspects worth mentioning though are:

- UPF: It is ok to fully describe the switching strategy, including the mapping to switch cells. DC-NXT will perform syntactical checks but does not insert switch cells. To utilize body bias supply set functions, such as "nwell" and "pwell", make use of:

```
set_design_attributes -elements {.} -attribute enable_bias true
```

- DFT: Minor additions to consider are during scan chain insertion: Make sure, the retention enable pins (Save/Restore), are set to a mode, where scan chains are traceable, e.g., constraining the retention enable pin accordingly in the dft file. Which value to set can be

retrieved from the liberty file, for instance.

```
set_dft_signal -view existing_dft -type Constant \
  -port [list {ret_en0 ret_en1 ret_en2 ret_en3}] -active_state 1
```

- DON'T-USE: For always-on buffering, it was confirmed by Synopsys, that DC-NXT requires both always-on buffers and always-on inverters, to be available as library cells. Otherwise, there will be no always-on buffering happening at all. That means, if you restrict the cell usage in your don't-use file, be aware of that fact.

4.2 Design Area Reduction in ICC2

In this chapter, the focus will be on how to drive placer and router into the desired direction, based on the given technology and IP, as well as explore some fundamental decisions with respect to certain power management considerations.

- Pin access settings and routing track alignments.
- Mixed preferred routing direction considerations.
- Primary power mesh alignment with bias nets.
- Control cell density under the power mesh.
- Auto density control and congestion effort handling.

These first level experiments will already consider additional overhead from secondary power mesh, retention flip-flops and bias routing.

4.2.1 Pin Access Settings and Routing Track Alignments

The specific architecture of this ultra dense std-cell library requires to have the lowest metal tracks aligned with the std-cell pins, in order to utilize available routing resources best. The std-cell documentation provides the necessary information and can vary across the various available cell architectures. Therefore, the purpose here is to create general awareness. Commands, such as those highlighted below are available to control the pin access behavior. Also note, that, desired track patterns require certain aspects to differ between the router technology files (*.tf), thus, it is recommended to choose that router technology file which fits to the main used std-cell architecture in the place&route block. In our case, GF provides those variations as part of the PDK collaterals.

```
set_wire_track_pattern -site_def <SITE_DEF> -layer M2 -mode uniform \
  -coord $M2_track_offset -space $M2Pitch -direction vertical

set_app_options -name route.common.derive_connect_within_pin_via_region \
  -value true

set_app_options -name route.common.connect_within_pins_by_layer_name \
  -value { …}
```

4.2.1 Mixed Preferred Routing Direction Considerations

Due to the ultra-dense cell architecture, the used std-cells come with a different preferred routing direction (in our case, a so called "HVH" scheme, Figure 2), compared to the memory macros ("VHV" scheme, also Figure 2). Therefore, some of the key aspects to be considered when setting up the place&route flow, are:

1. Changing preferred routing direction over the memory macros, for reasons, such as avoiding fringe caps, and to avoid conflicts with the macro's abstract structure, such as the pg-pins. The below approach of how to utilize routing guides, has been proven to be very successful, also in the past on other designs with a similar constellation:

```
create_routing_guide -switch_preferred_direction -layers { M5 M6 } \
  -river_routing  -boundary {<Memory-Macro-Boundary>}
```

2. Power mesh separation between memory macros and std-cell area to follow the pin-layer structure of the macro.

Note:

There are memory compilers available with the option to enable a flipped preferred routing direction scheme, such as, changing signal-pins layers (e.g. form M2 to M3), and, for avoiding fringe caps, adding a signal-routing blockage to avoid routing in non-preferred direction (from the macro's perspective) of the first available routing layer on top of the macro (M5 in our case). That blockage would still allow pg-objects on that layer.

We also would like to refer to a SNUG paper from 2022 [9], which is dedicated entirely to this topic.

4.2.2 Primary Power Mesh Alignment

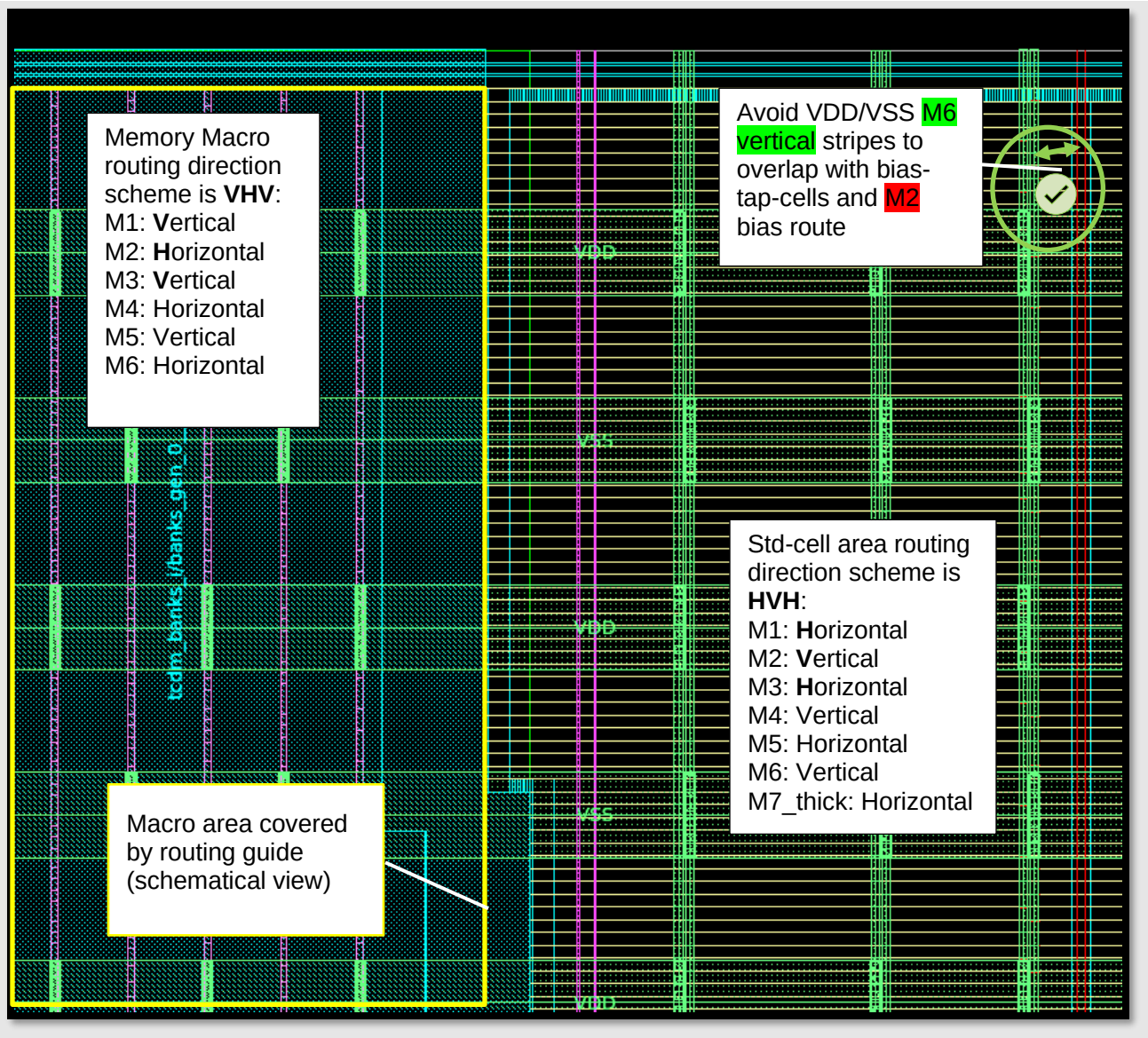


Figure 2: Different routing direction on top of std-cell area (right, HVH) and on top of memory macro (left, VHV). Additionally visible are bias-tap-cells and boundary cells.

To avoid clashes between the primary power mesh and, between body bias routing, certain alignment is required. The above green circled area is enlarged in the picture below:

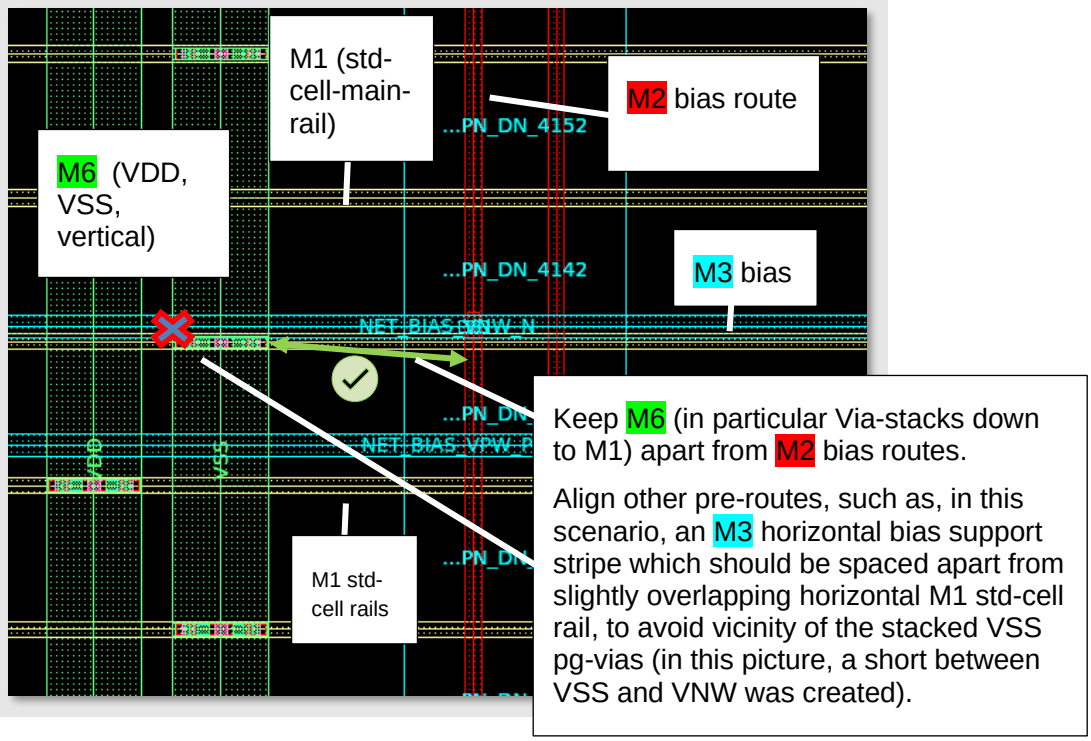


Figure 3: Spacing M6 (VDD, VSS, green) from M2 (VNW, VPW, red), and M3 (VNW, cyan) from M1 (VSS, yellow)

There is clash between horizontal bias route in M3 (cyan) and VSS vias stack from M6 (vertical, green) down to M1 std-cell main rails (horizontal, brown). An offset in y-direction is required here. Also, the M6 vertical stripes need to avoid the M2 vertical bias routes, connecting down to the bias tap cells.

4.2.1 Controling Cell Density Under Power Mesh

Initial trials have unveiled that, underneath the M6 vertical mesh, the cell-density is too high.

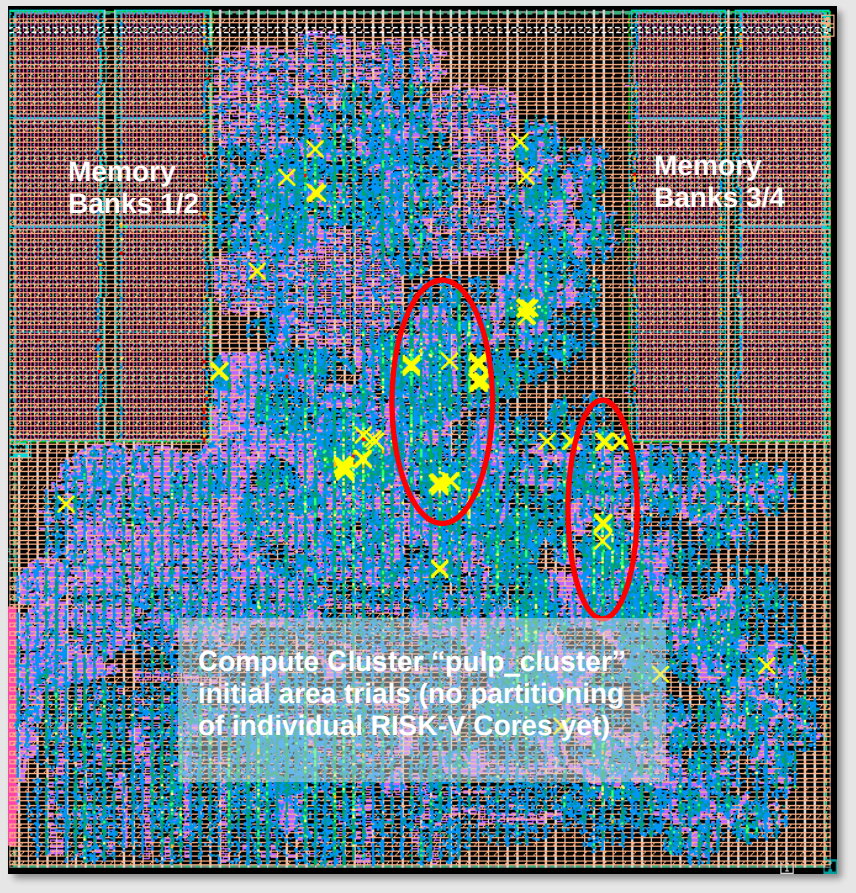


Figure 4: Congestions below vertical M6 power and ground stripes (116 DRCs after route_opt)

The following two app-options are a great way to control this and improve the overall congestion situation:

```
set_app_option -name place.common.pnet_aware_density -value 0.40
set_app_option -name place.common.pnet_aware_layers -value {M6}
```

Not guiding the tool to control density below M6 stripes resulted in congestion and DRCs.

4.2.2 Auto Density Control and Congestion Effort Handling

With newer versions of ICC2 (and FusionCompiler), the automatic density control has been enhanced significantly. Based on trial experiments, and with great consultancy by Synopsys, the following two options have been identified to be most effective in order to optimize the std-cell distribution:

```
set_app_options -name place.coarse.auto_density_control \
   -value enhanced ; # default meanwhile
```

```
set_app_options -name place.coarse.max_density -value 0.4
```

These settings converged after a couple of guided trails, together with Synopsys. One key measure was to compare actual settings with the tool's progression during place_opt phase, by looking out for PLACE-027 messages in place_opt log file.

Example first occurrence of the message:

```
Information: Automatic density control has selected the following settings: max_density
0.48, congestion_driven_max_util 0.87. (PLACE-027)
```

Example last occurrence of the message:

```
Information: Automatic density control has selected the following settings: max_density
0.48, congestion_driven_max_util 0.89. (PLACE-027)
```

By examining the log file, a conclusion was also, to leave the following app option unspecified (default) and let the tool auto-converge: `place.coarse.congestion_driven_max_util`.

In the appendix, we did put some experiments side by side to allow the reader to follow our journey.

A note on std-cell utilization: Although the above appears to concentrate on the utilization in the first place, the goal is not to achieve the highest utilization per se, but to achieve the smallest routable area. Utilization is one factor, but arrangement of the std-cells to support the chosen metal stack is equally important. Std-cells, which are highly compact, might look less attractive from a utilization standpoint in the first place, but will win the area battle at the end, if the tools are guided properly (in the scope of overall application considerations).

4.3 Low Power Techniques

One specified goal is to enable shut-down of individual RISC-V cores to save power (leakage) if a core of the Compute Cluster is not used. Therefore the 4 RISC-V cores are implemented as individual power islands. Power gates are inserted into the design (in the following referred to as core power switches) as a means for shutting down the individual RISC-V cores. A buffering scheme which is always on needs to be provided to control the core power switches. The individual cores should not be implemented as hard macros but treated in a flat approach within the "pulp_cluster" design.

Control registers of a small sub-hierarchy of each switchable RISC-V core are mapped to retention-flip-flops, allowing them to retain their state, while the rest of the core is switched off. Therefore, additional always-on buffering to the save/restore pins of these retention-flip-flops is required.

For the instances in the TCM of the Compute Cluster an SRAM compiler type is used which provides built-in shut down features. Therefore, there are no additional external power switches needed for those memories, which is an overall area and power saving. Instead, a power down control pin is provided for each SRAM instance which can be connected separately or combined to a power control module (not part of the design).

For the standard cell logic as well as for the SRAM instances adaptive forward body bias (ABB) is implemented to allow for a reasonable high performance in slow delay corners. This means that related body bias power nets need to be routed.

The specifications mentioned above drive the decisions for power mesh strategies which have to be selected.

The power intent for always-on and switchable domains is illustrated in Figure 5. Ground nets and bias nets are common for all components in the design and are therefore not displayed.

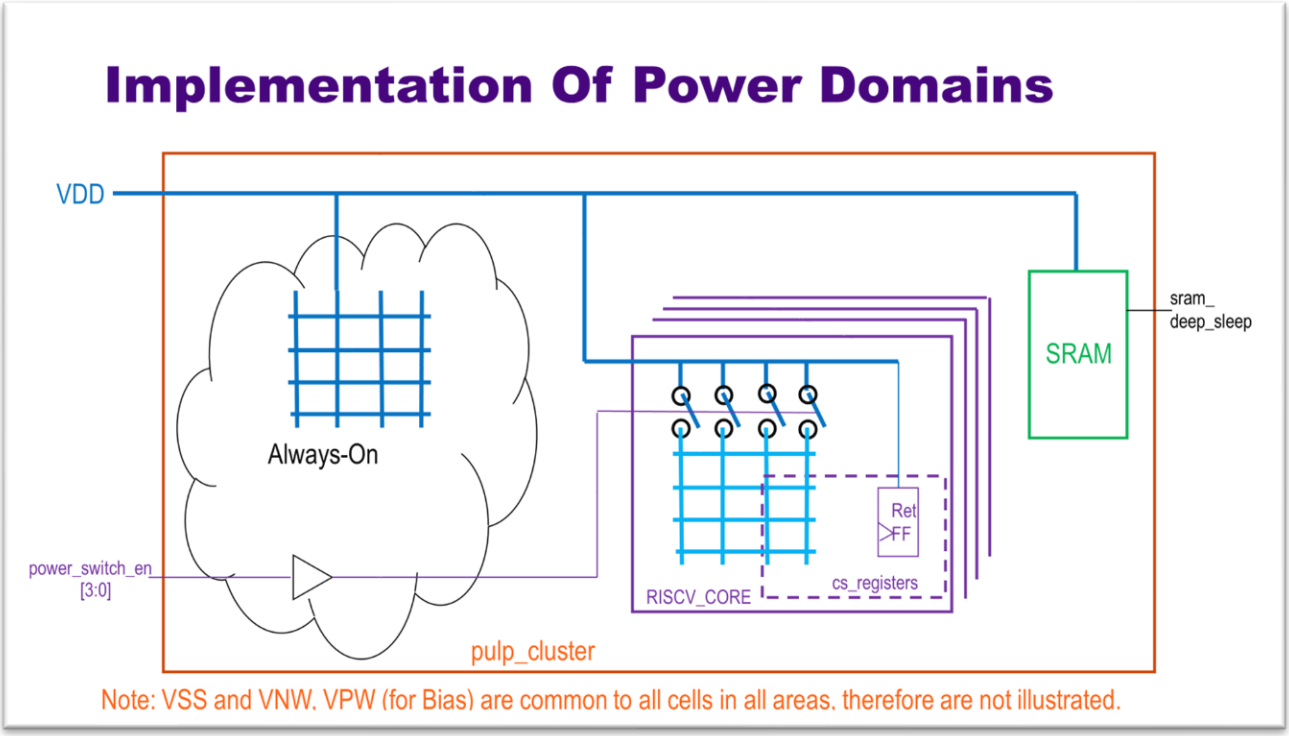


Figure 5: Representation of "pulp_cluster" power intent.

UPF is used to describe the entire power intent from start of synthesis throughout the design flow. This simplifies the implementation of even such a complex power structure because native tool commands can be used and work seamlessly. UPF based implementation tools understand how the various multi-voltage cells, such as switches, always-on buffers, retention-flip-flips and isolation-cells, with their various pg_pin types (such as "primary_power", "backup_power", "always_on") have to be connected to the respective supply nets, where standard cells of different power domains are allowed to be placed (driven by the multi-voltage aware flow, via power intent and `set_voltage` commands), and the insertion of the required support cells (e.g., always-on buffers, isolation cells). The interested reader may also refer to the SNUG World 2021 presentation video [8].

Power routing which does not follow a mesh approach (e.g. retention supply for distributed retention flops) can be automatically handled by ICC2 during signal routing, or by utilizing an additional supporting small always-on mesh. This topic is touched on again in chapter 4.3.1 .

Remark: Focus of this paper is a block level implementation. In the SoC scope, additional aspects of combining ABB with power gates have to be considered, such as proper power up and power down sequencing. Please refer to [1] for additional details.

4.3.1 Floorplaning of Voltage Areas

Figure 6 illustrates the final floorplan of the top level of the “Compute Cluster” design named “pulp_cluster”. The 4 RISC-V modules are placed as 4 bounds in the bottom left and bottom right areas. These 4 bounds are used as voltage areas later, resulting in a total of 5 power domains (4 switchable for RISC-V and 1 always-on for pulp_cluster).

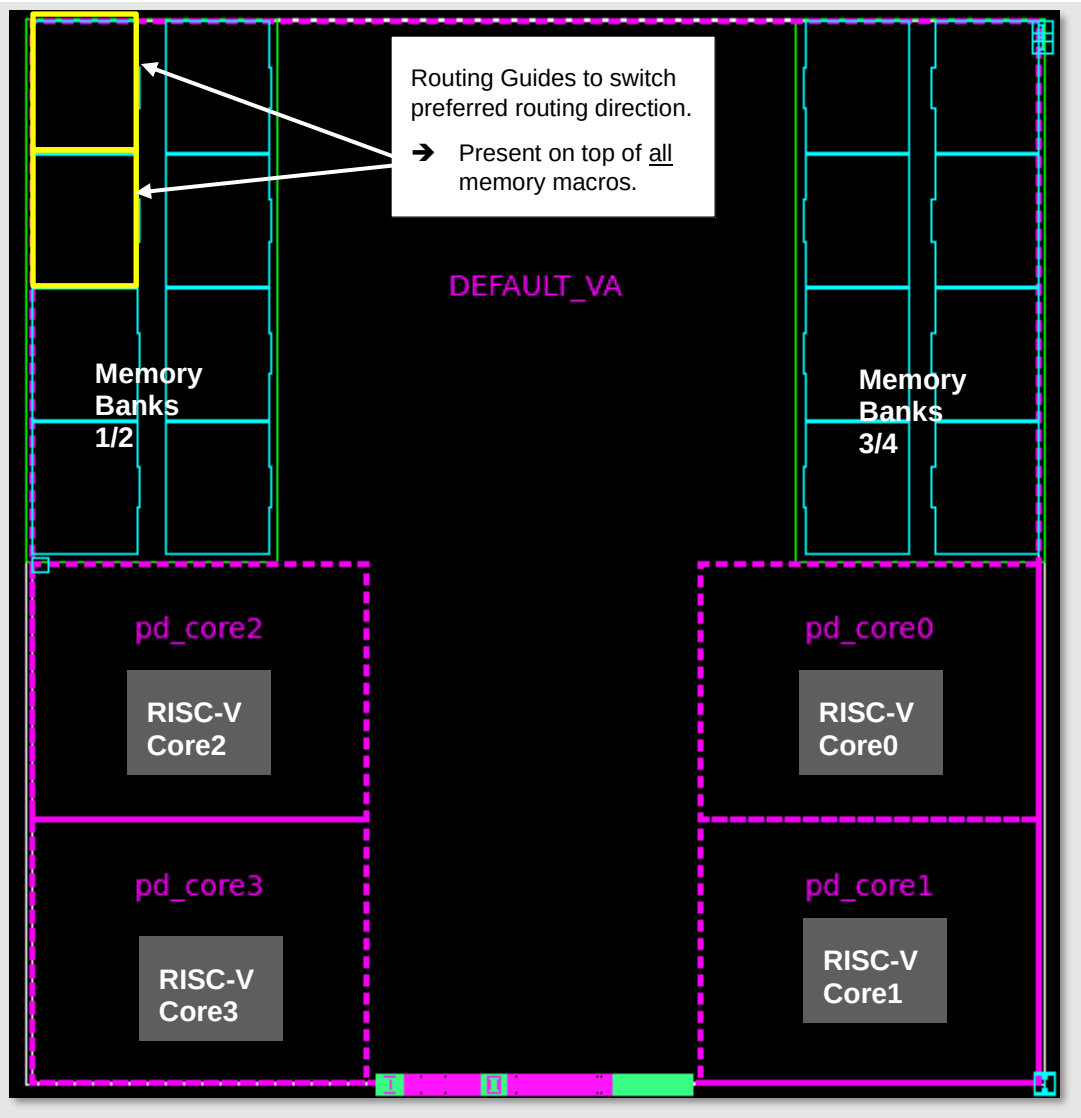


Figure 6: Final floorplan.

The 2x4 instances of SRAM in each of the top left and top right areas represent the entire TCM which is shared between all 4 RISC-V cores via a crossbar structure (not separately shown). All IOs of the “pulp_cluster” are located along the bottom center of the block. Early place and route experiments show that with this floorplan timing can be met relatively easily; on the other hand, this rather simple floorplan could be laid out also to have the pins on one of the vertical edges if next higher design hierarchy level requires it.

This floorplan was developed using early place and route trials showing that critical timing paths in

particular from the SRAMs to the cores (Figure 7) could be met and routing could be closed. The targeted std-cell utilization for the individual voltage areas was derived from these early trials, too, towards an overall optimal placement density (Figure 8). Note, the cell distribution as shown in Figure 7 below does not yet reflect the final floorplan.

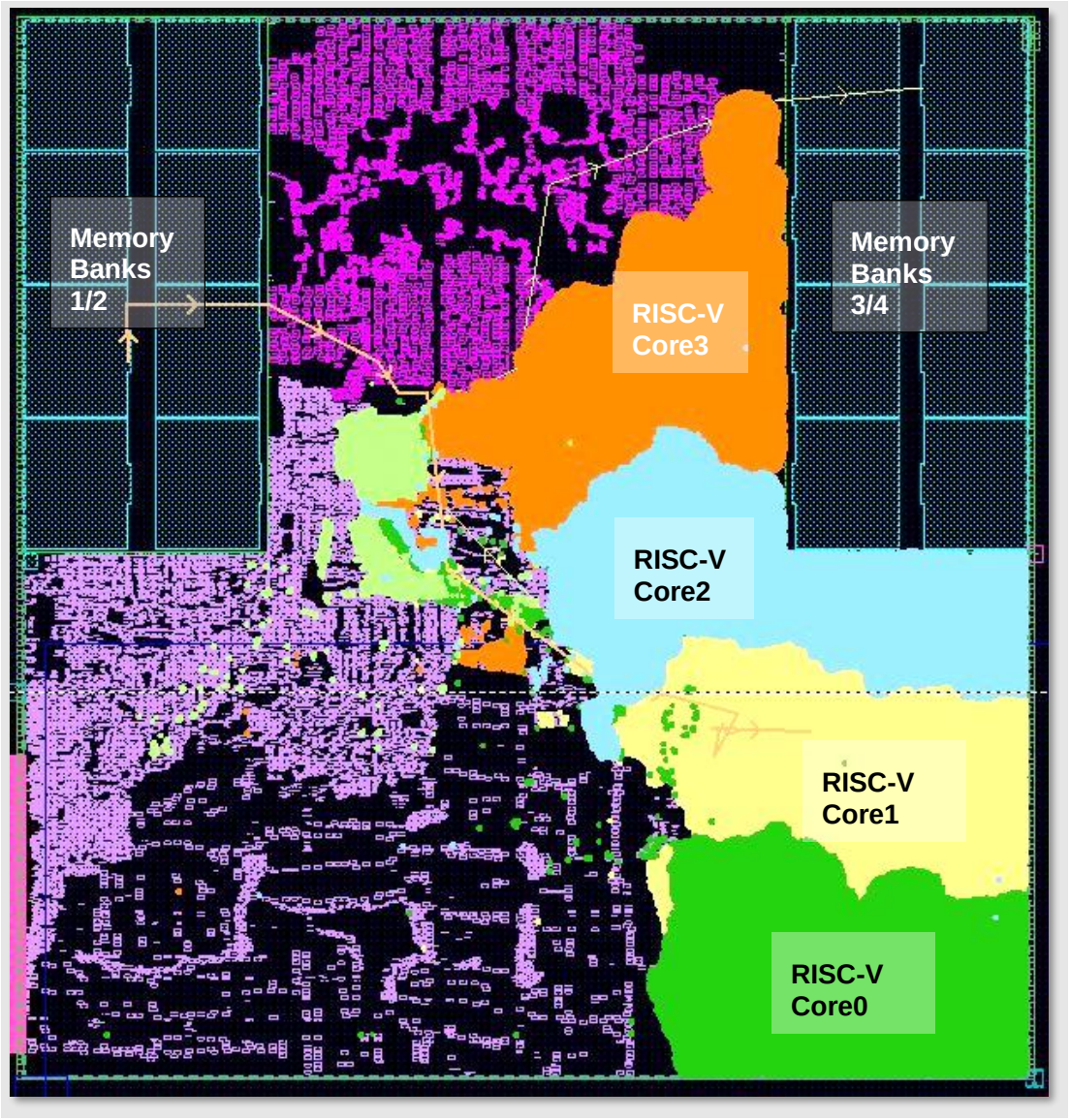


Figure 7: Early placement trials to understand logic hierarchy distribution (no switchable domains exist yet).

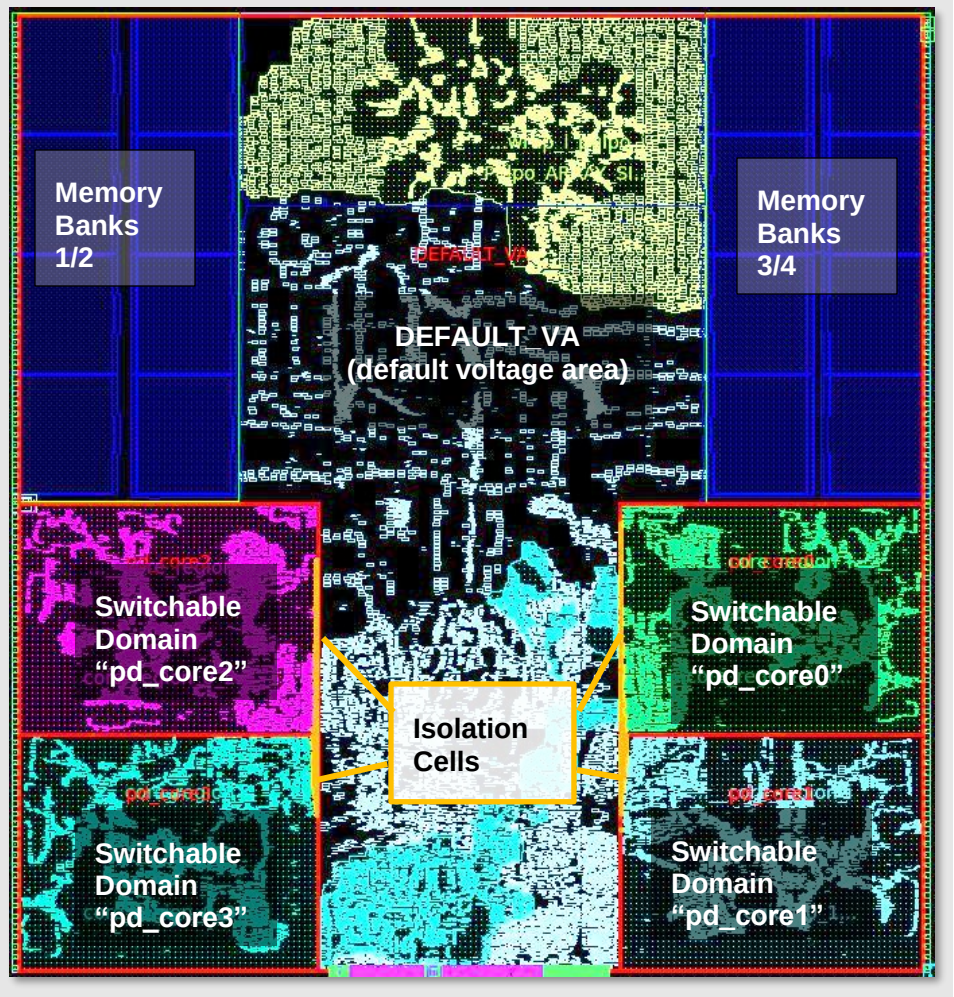


Figure 8: Final floorplan with refined voltage areas shapes, and isolation cell location.

Boundary cell and Tap cell insertion (a requirement in this technology) is now controlled per voltage area. It turned out to be beneficial to first execute the commands for the switchable voltage areas, followed by another round, specifying the default voltage area:

```
compile_boundary_cells -voltage_area [get_voltage_areas pd_core*]
compile_boundary_cells -voltage_area [get_voltage_areas DEFAULT_VA]
create_tap_cells -voltage_area [get_voltage_areas pd_core*]  …
create_tap_cells -voltage_area [get_voltage_areas DEFAULT_VA] …
```

A snippet of the resulting boundary cell and tap cell placement can be seen in Figure 9. Note that the initially chosen guard-band between voltage areas can be shrunken further.

4.3.2 Isolation Strategy Considerations

Initial experiments showed that providing the placer with more flexibility for where to place the isolation cells leads to better results. So, the approach was switched from "destination" cells (std-cell main rail = VDD: always_on) to "source" cells (std-cell main rail = VDD0...3: switchable), by changing "`set_isolation_strategy -location parent`" to "`-location self`". Dual-rail isolation cells with an additional always-on supply are required in this case, which the technology provides.

4.3.3 Transitioning from Regular Power Mesh to Gated Power Structure

A major piece of design work in the present paper is the decision making for the power structure, namely the power mesh configuration of the pieces which are switchable by the power gates. The following section describes the details behind this process. In this paper, we will concentrate on the implementation part and will not touch on system level aspects, such as adding a controller to the RTL for handling the power management.

Ground (VSS) and Bias (VNW, VPW) are common for all domains, therefore do not have to be changed. This means switching from an always-on supply (VDD) to a switchable supply (VDD0...3) by utilizing as less as required additional resources is the goal for the present design.

The start point is a regular power mesh which has the standard cells connected to horizontal M1 (metal1) rails. These M1 rails are connected to vertical M6 (metal6) straps using specific VIA structures at each intersection of M1 and M6 as illustrated in Figure 9 . This means that each M1 rail gets its own direct connection to an M6 strap. The M6 straps are connected to the always-on power named VDD here. Proper EMIR analysis shows that this kind of power mesh fulfills all IR drop and electro migration requirements for the present design.

While, in the always-on domain, both, VDD and VSS are connected down to M1 std-cell rails (Figure 9 top right snippet), in the switchable domain, only shared VSS is connected down (Figure 9 bottom right snippet), and, VDD will be connect to the input supply of the power switches first (Figure 11).

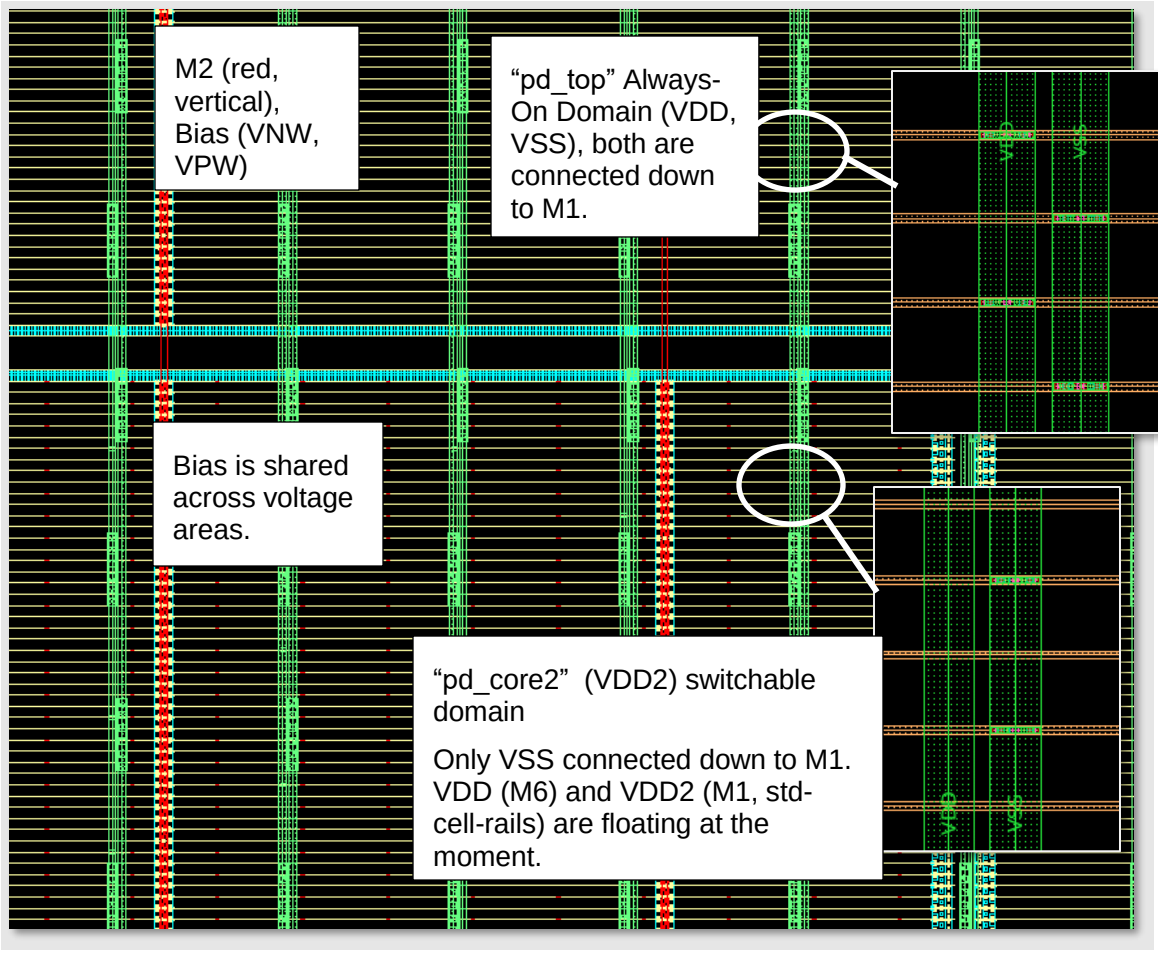


Figure 9: Power mesh transition from always-on domain to switchable domain “pd_core2” (only M1 to M6 turned on, no power gates, no M7_thick)

With the further introduction of power gates (Figure 10 and Figure 11) the VIA-based connection between M6 and M1 is interrupted (already sketched in Figure 9, bottom-right zoom-in snippet). The connection between VDD (always-on) and VDD0…3 (switchable) is going to happen as a switchable connection by utilizing so called power gates or power switches (in the present technology called PGAT cells).

The first steps of the planning process are to decide which power cells to use and which metal layers to use to achieve the desired connectivity. For the present paper it was decided to use the PGAT cell of which the structure is shown in Figure 10.

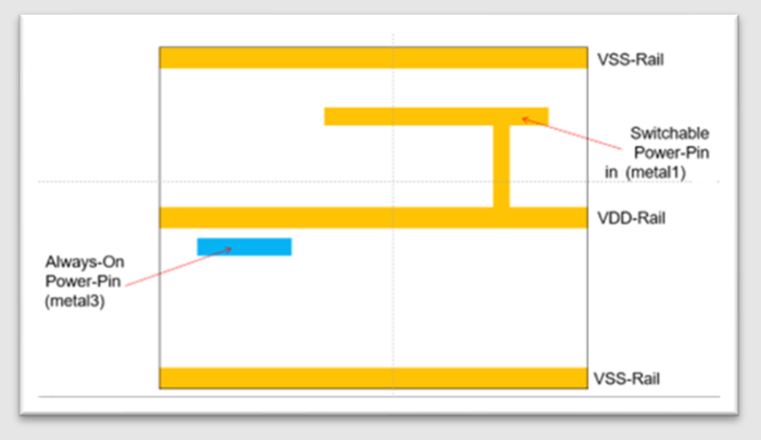


Figure 10: Schematical representation of power gate cell (PGAT)

As illustrated in Figure 10 the PGAT cell contains a power input pin in M3 (metal3) to be connected to the always-on power domain VDD and another power pin in M1 (metal1) as an output to be connected to switchable power domain VDD0…3. Assuming that the same vertical M6 straps are kept as used in the regular mesh, the connectivity is done using VIAs to drop from M6 straps down to M3 power switch cell pins on the input side (see also chapter “Tips and Tricks”) and aligning the power switch cell M1 pins to the M1 rails on the output side.

While adding a power switch cell to each intersection between M6 straps and M1 rails would lead to the same kind of mesh structure as with the regular mesh, and hence similar EMIR results, this would mean a tremendous number of additional cells in the design added in columns at the pattern of the vertical M6 straps, that is, a major loss of placement area for logic gates (not mentioning the additional leakage introduced by the power switch cells). Therefore, a smarter way of organizing the use of power switch cells has to be found. In the present paper the decision was made for a checkerboard pattern of power switch cells as illustrated in Figure 11.

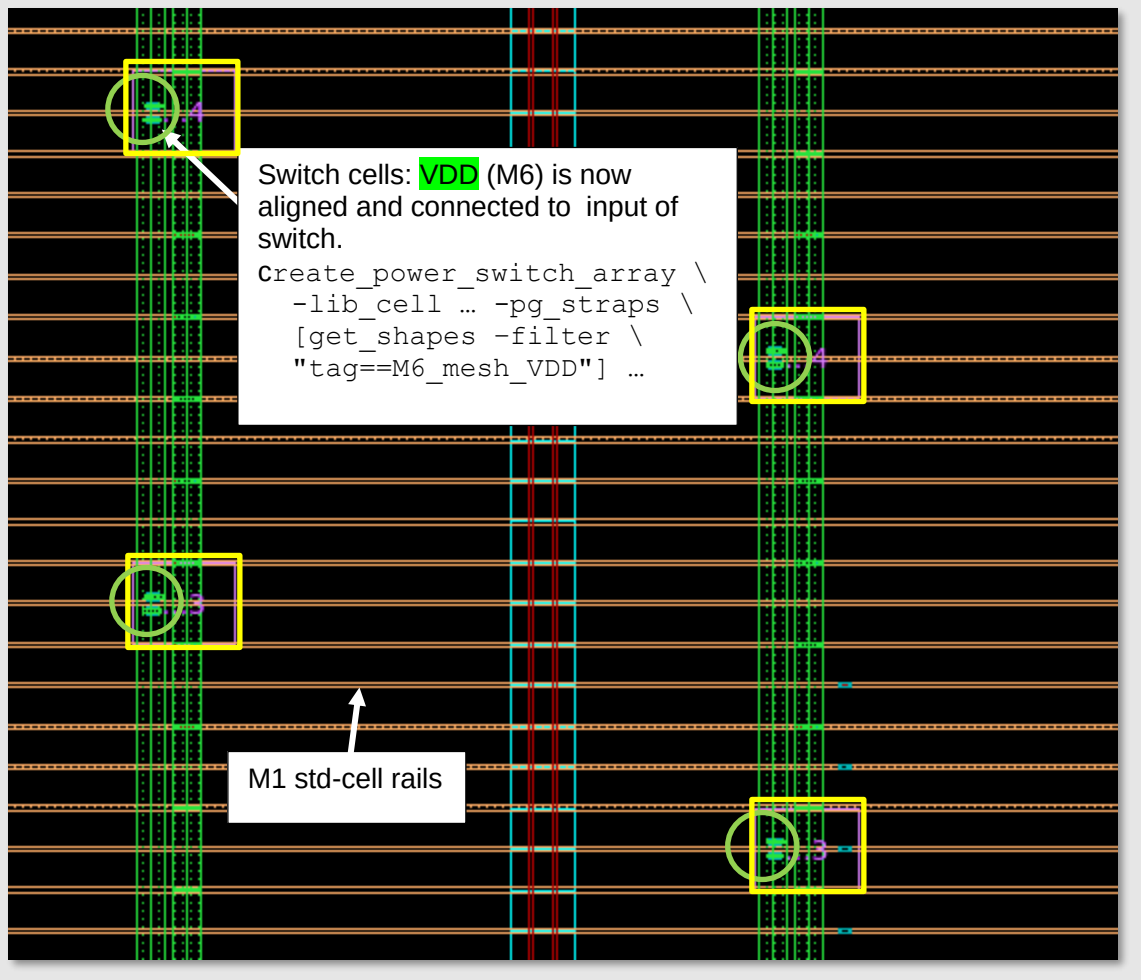


Figure 11: "pd_core2": Power mesh with switch cells in checkerboard pattern.

Obviously, the chosen structure leaves a bunch of M1 rails without connections. This means that additional straps need to be placed to build the complete connectivity and also to fulfill EMIR requirements of the design. Two further steps are taken in the present paper to obtain the final mesh structure for the switchable power domain VDD0…3 as illustrated in Figure 12. The first step is adding vertical straps in M4 (metal4) such that VIAs can be placed at each intersection with the M1 std-cell-rails, and they all get connected to each other. These additional vertical M4 straps are also aligned with the placement of the power switch cells, such that VIAs can be placed directly at these switch cells. The second step is placing horizontal M5 straps to avoid all that current in the horizontal direction which needs to be carried just by the M1 rails – usually the M1 rails are the parts of the power mesh with the highest resistance and therefore the weakest links.

The way to calculate the widths and pitches of the different power mesh elements can vary from best guesses to paperwork calculations depending on data which is readily available to the designer. An important aspect of this step is a proper alignment between different components, like TAP cells which are used for body bias voltage distribution, power switch cells, the different power mesh elements, etc. to avoid shorts and other design rule violations. Working with a multiple of std-cell site-width and std-cell track-height turned out to be very supportive.

In the end a thorough EMIR analysis needs to be performed to prove the robustness of the generated power mesh structures. Of course, it is highly recommended to perform such analysis as early as possible in the design process.

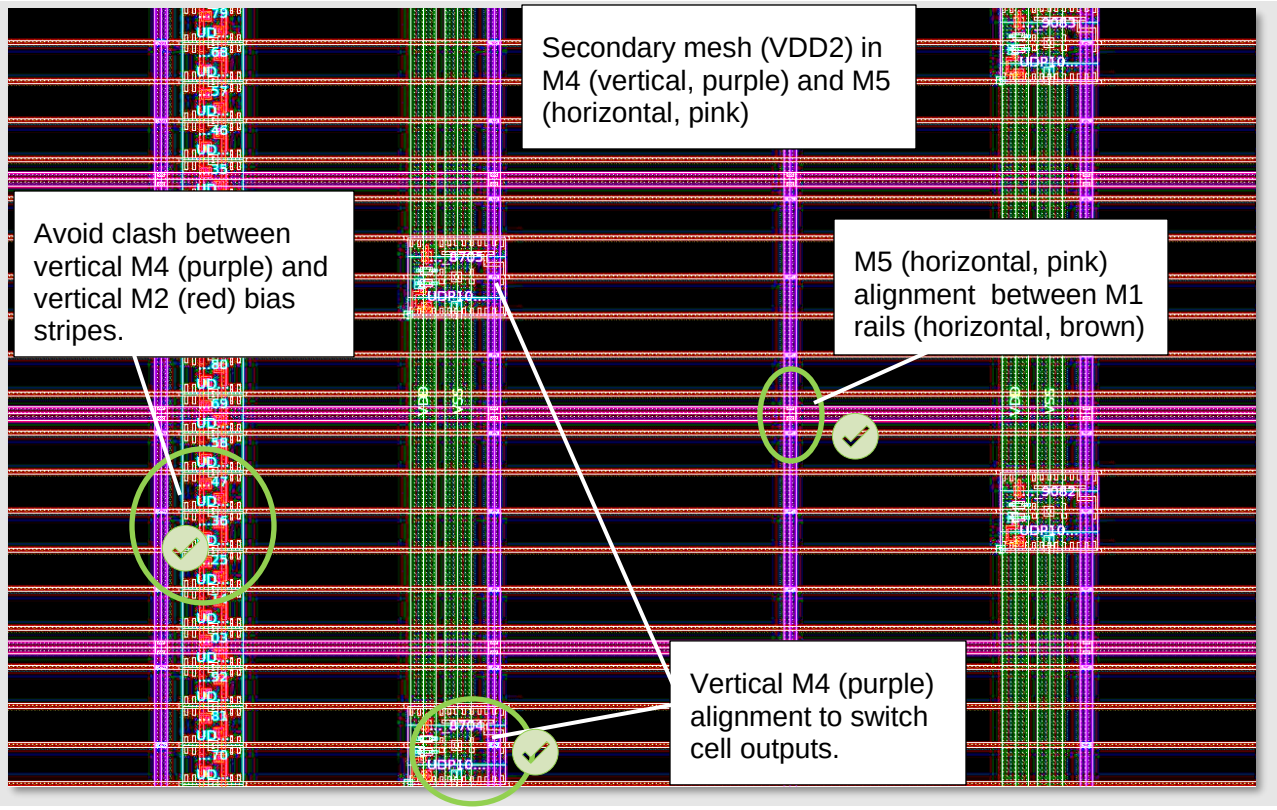


Figure 12: "pd_core2": Power mesh with power gates and support in layers M4 and M5

4.3.1 How to Connect Always-On Power of Dual-Rail Cells in the Switchable Domains

In this chapter, two ways will be discussed, how to connect dual-rail cells, such as retention-flip-flops, always-on buffers, isolation cells, to the always-on supply:

1. The first approach is, to just let the signal router do the job (default).
2. The second approach is to create a small additional always-on supporting mesh which overlays directly with the always-on supply pins of the dual-rail cells (if the cell-architecture supports this).

Both approaches will be demonstrated with focus on the retention-flip-flops present in each of the four switchable domains.

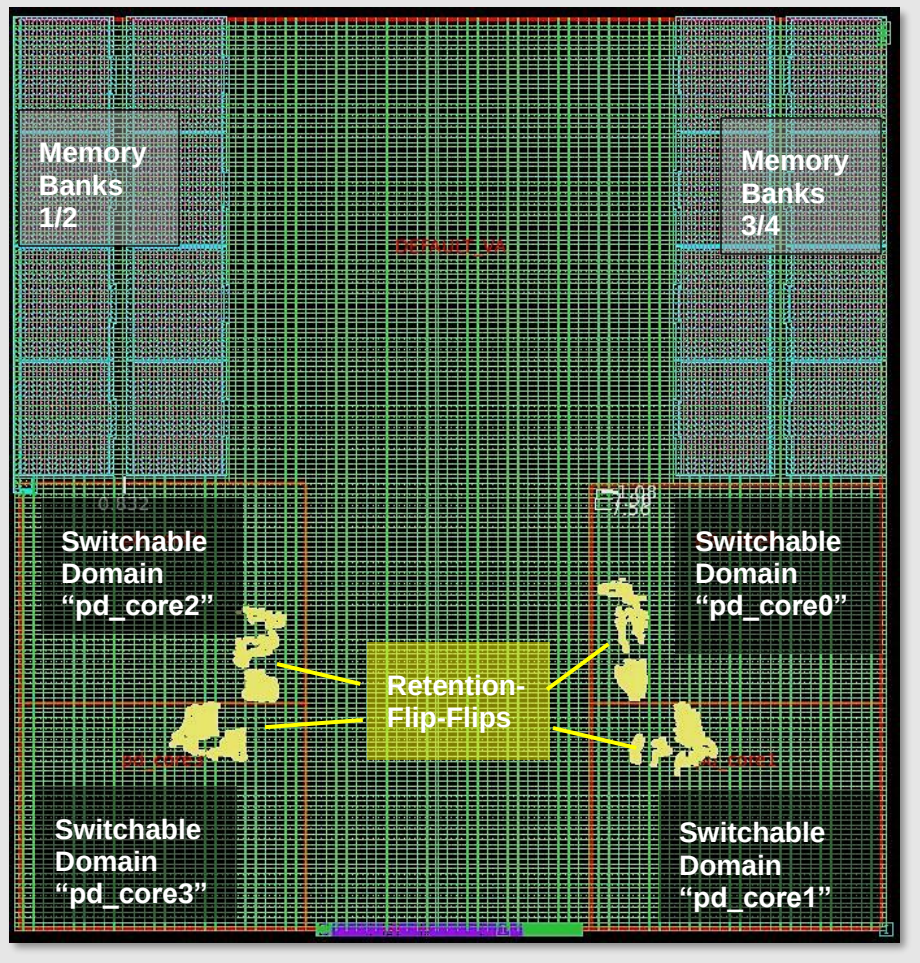


Figure 13: Location of the retention-flip-flops within each voltage-area.

First approach: Let the signal router do the job:

Below picture shows, how the retention-flip-flops' VDDR pin (retention power, always-on) is connected by the signal routing automatically, using default options, to the closest always-on M6 stripe.

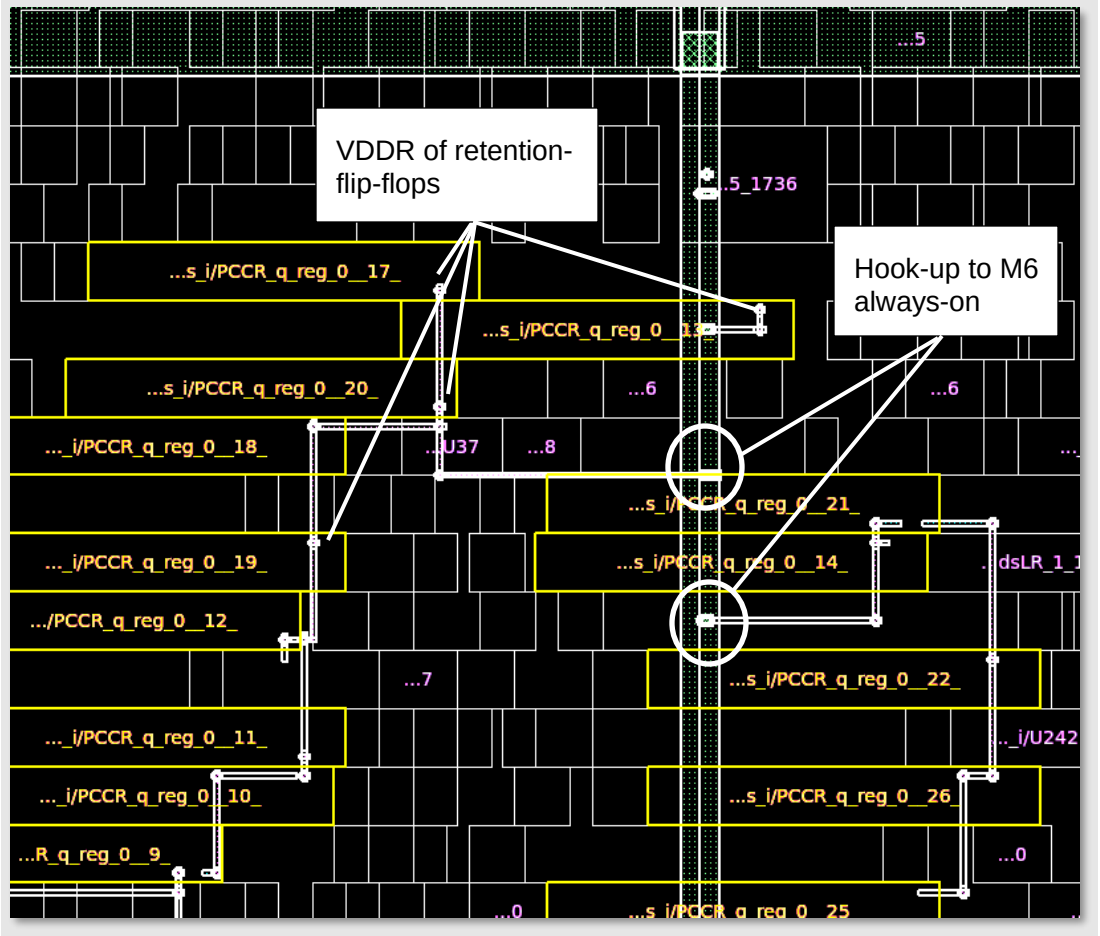


Figure 14: Retention-flip-flops' VDDR (always-on retention supply pin) power connection.

Second approach: Have an additional small always-on supporting mesh, if the std-cell architecture supports this:

Always-on power pins of all the std-cells used are laid out using the same layer (M3) and the same y-offset. That allows to create a horizontal M3 mesh (VDD, always-on), which is aligned with the position of the M3 VDDR always-on pins. A physical connection is automatically made by doing that. Either, the entire switchable domain is spun by those horizontal M3 stripes, or, only across the desired cells and rows. Figure 15 illustrates such an approach.

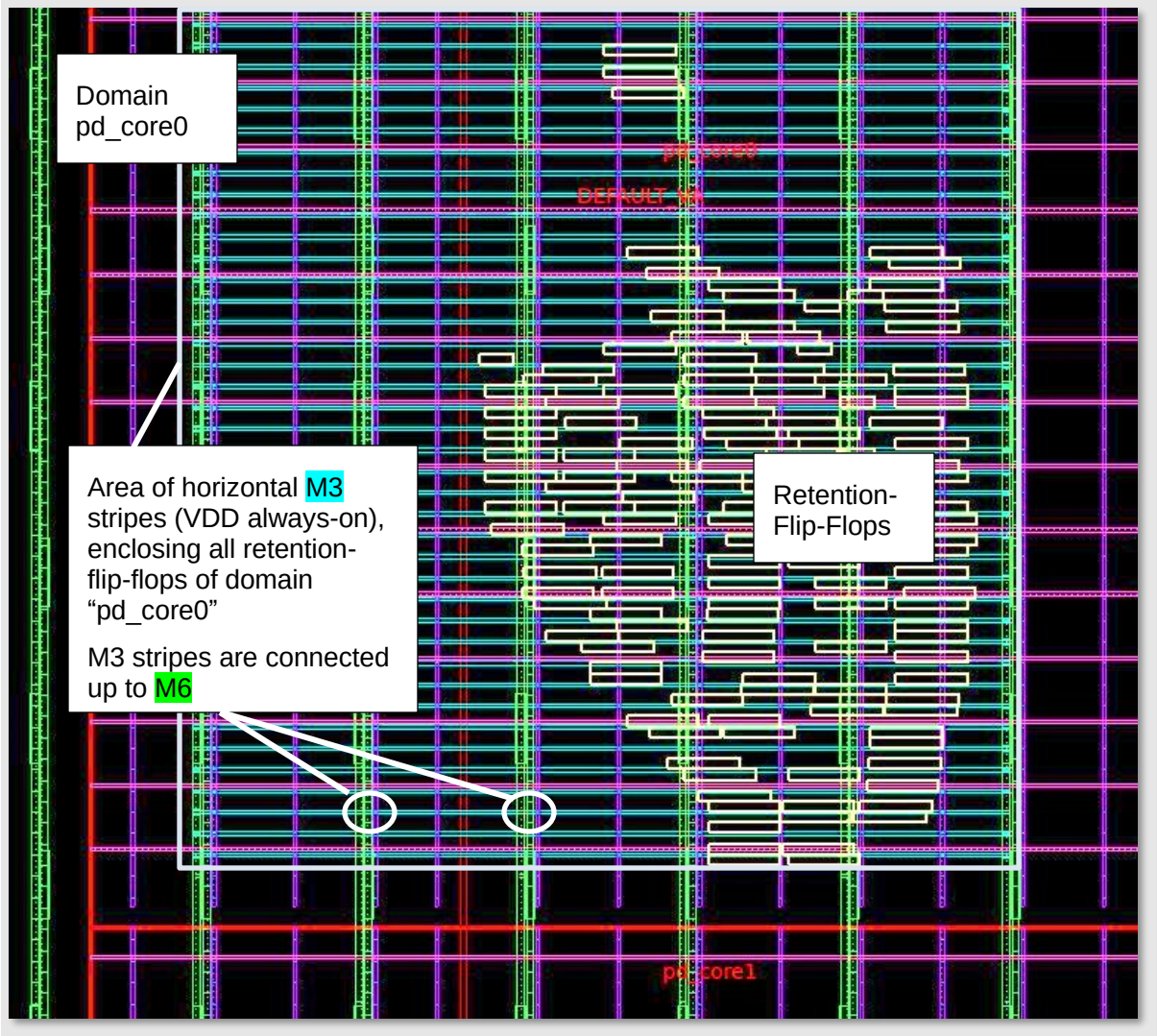


Figure 15:Snippet from domain pd_core0: Horizontal M3 stripes serve as always-on connection to VDDR pins of the retention-flip-flops

Figure 16 illustrates how the M3 stripes are aligned with the VDDR pins. Some details of the std-cell layout are hidden for confidentiality purposes.

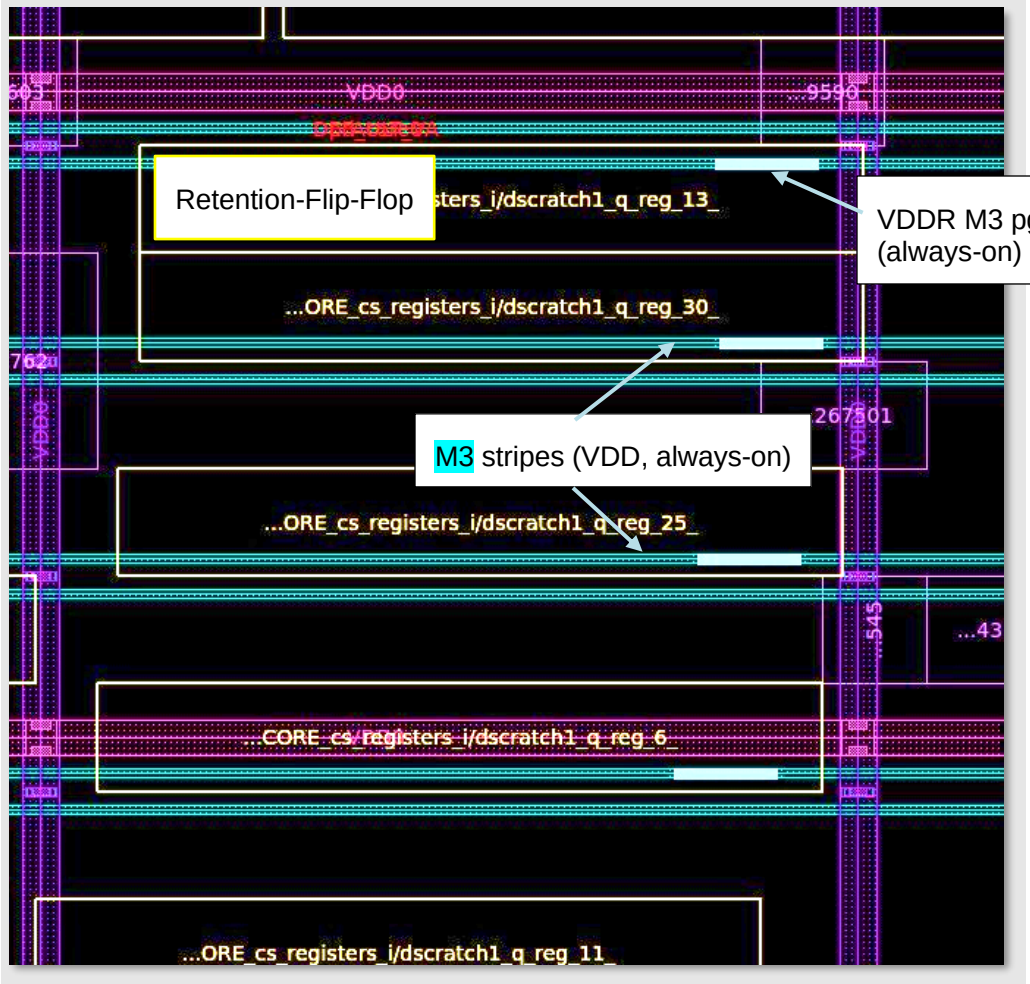


Figure 16: M3 stripes (VDD, always-on) are aligned with M3 VDDR pins (always-on). Note: Visibility of M1 std-cell-rails is turned off.

This approach is compatible, not only with the entire dual rail power library, but with the complete std-cell set, which is optimized for this approach.

Note: it is even possible to span the entire switchable domain with such a supportive always-on mesh structure.

Also, combinations of the above demonstrated approaches are possible.

Which one of the above discussed is the more efficient approach depends for instance on the number of dual rail cells which need a connection to the always-on supply. If there are just a few cells, signal router is most probably the best choice. Otherwise, having pre-routed always-on stripes in M3 could be more efficient. It may also depend on the number of other dual-rail cells, such as always-on buffers, which will be inserted during optimization.

5. Issues

5.1 STAR 5352238 – “DC printing bogus messages during the RTL gate restructuring”

No issue per se but confusing. Messages similar to below are printed, which are not true. All operating conditions are present.

```
Cannot find equivalent cell for <dual rail inverter> for operating
condition <operating_condition>
…
```

5.2 STAR 5346925 – “DC-NXT behaves unexpectedly by picking Dual rail cells”

When not correctly put on don’t-use, rarely, dual rail clock gates are inserted in the always-on domain, instead of single rail clock gates. Later, check_mv_design correctly errors out. As we do not need dual rail clock gates, we put them strictly on don’t-use.

5.3 Case 01585702 - STAR Filed - “M3 VDDR pins can’t be legalized under M3 VDDR rail” - ICC2

In ICC2, on newly added dual rail cells during optimization (e.g., place_opt, clock_opt*, route_opt), which are not “pg-connected” yet, the legalizer does not treat the M3 strips as pg-network, an thus attempts to legalize them outside of the vicinity of the M3 stripes, which is an impossible attempt → QoR issues, huge runtimes; Messages like: “`Warning: After 960000 legalization trials, instance CORE_3__core_region_i/CL_CORE_RISCV_CORE_cs_registers_i/place_opt_ZBUF_489_inst_230642 (<dual rail buffer>) has the following legality failures: (LGL-164) … pg_drc  959904  99.99%  100.00%`”

Workaround: Put the M3 VDDR mesh after place_opt. Also, consider restricting this structure to the span where it is most efficient. Subsequent updates during clock_opt_* and route_opt stages rarely run into the above issue any more.

6. Result

6.1 ICC2 Routing and Timing

Both of the described secondary power techniques in chapter 4.3.1 lead to a routable design, which is timing clean (not looking at minor violations). Figure 17 shows the routed database with a handful of routing DRC violations left. The cell density map, together with the Global Route congestion map suggest that there is still room for further reduction in the overall area. Also, the guard bands between the voltage areas and to the always-on domain were sketched rather conservatively which allows further improvement.

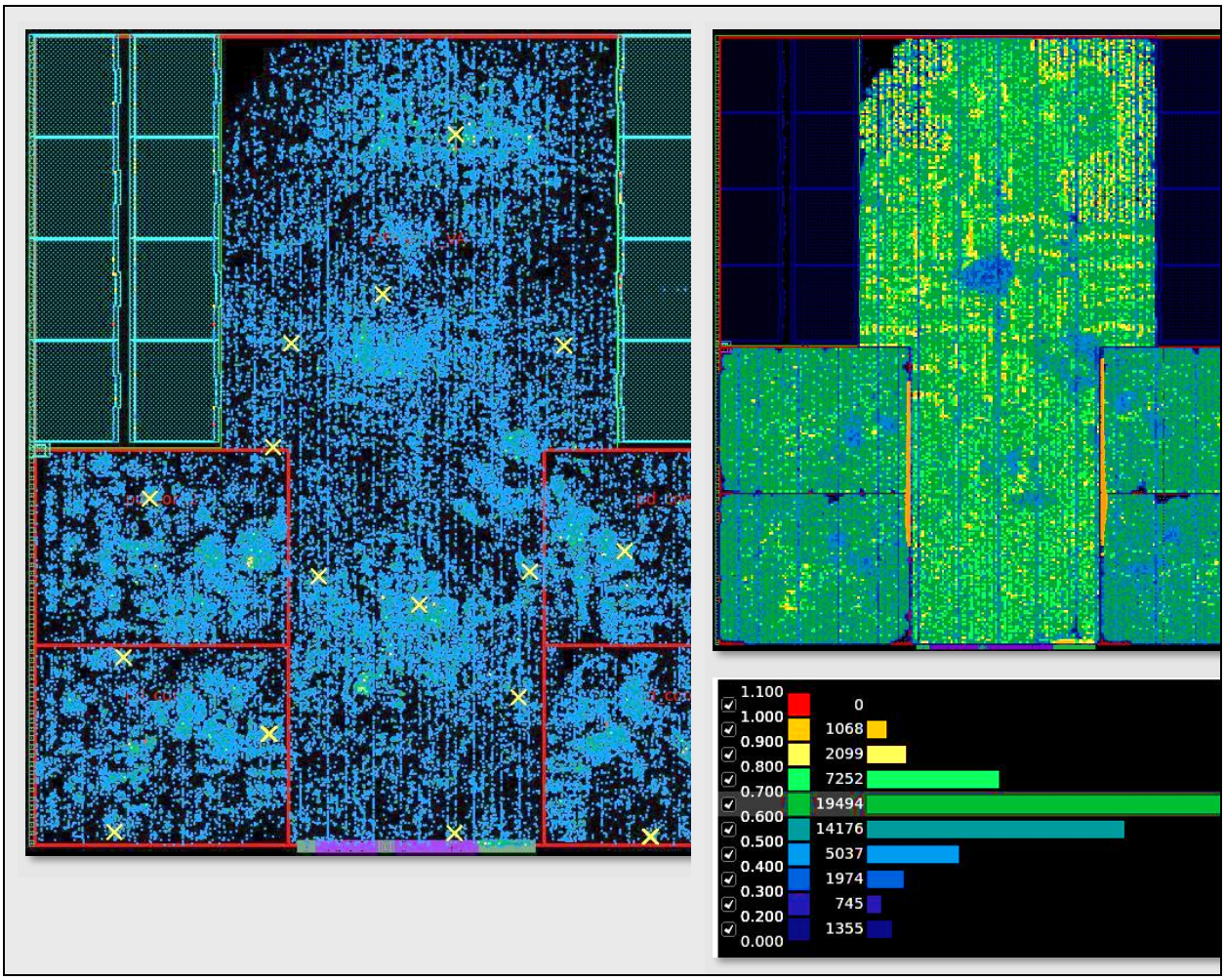


Figure 17: Global Route, DRCs (left), Cell Density and location of isolation cells (right); final run w/ switchable domains.

Total number of DRCs = 23 (after route_opt)

Timing Qor (after route_opt)

- Setup (WNS, TNS, NVE): -0.011ns, -0.019ns, 5
- Hold (TNS, WNS, NVE): -0.002ns, -0.005ns, 6

At the time of compiling the data for this paper, we have yet to collect results from STA. However, we are not expecting major surprises based on our experience with flow correlation between ICC2 and PrimeTime in this technology.

6.2 EMIR Analysis with In-Design Redhawk

IR-Drop and Electromigration Analysis is done with In-Design Redhawk. Power pin locations are inherited from the top-level redistribution layer and the power and ground bump locations. Power switch models were provided with the std-cell IP and passed over via a local “gsr” file, together with additionally required signoff-settings.

```
analyze_rail -nets {VDD VDD0 VDD1 VDD2 VDD3 VSS} -voltage_drop static \
   -electromigration -extra_gsr_option_file ./ICC2/RH/pulp_cluster.gsr
```

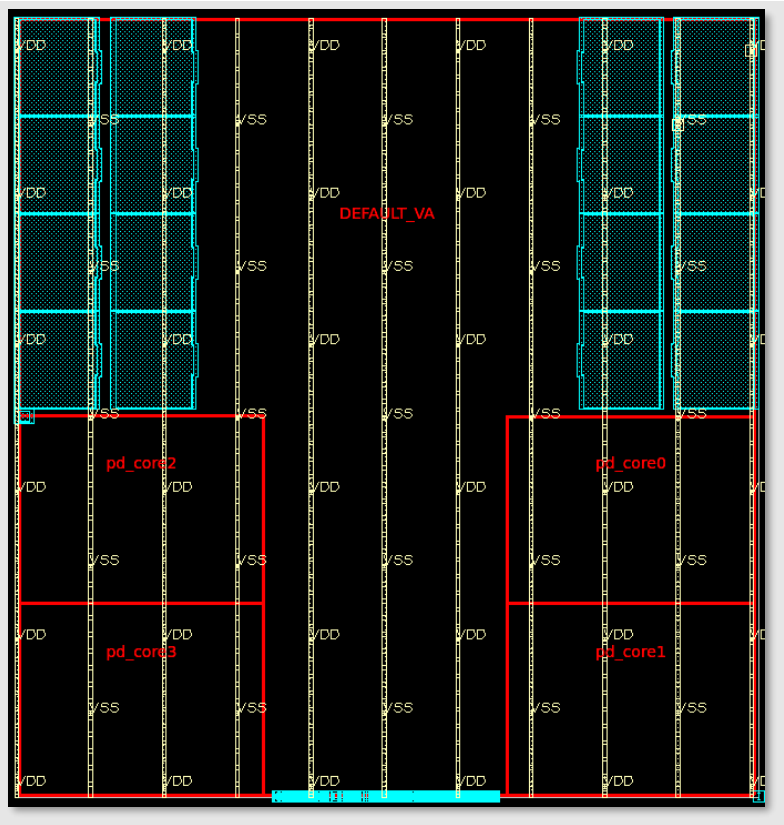


Figure 18: Power and Ground injection from top-level bumps via redistribution layer.

Static IR-Drop and Electromigration Analysis did not show any major issues, except a local weakness in the always-on supply to a group of isolation cells. We did not specify anything, so the router has freedom to cluster the number of secondary pins to be hooked up to an always-on pg-mesh object. This obviously led to a weakness (Figure 20).

As a counter measure, either a mesh approach described in Figure 15 and Figure 16, may be chosen or, the maximum allowed secondary pins clustered together for the hook up may be adjusted. This can be controlled with the following app option:

```
route.common.number_of_secondary_pg_pin_connections
```

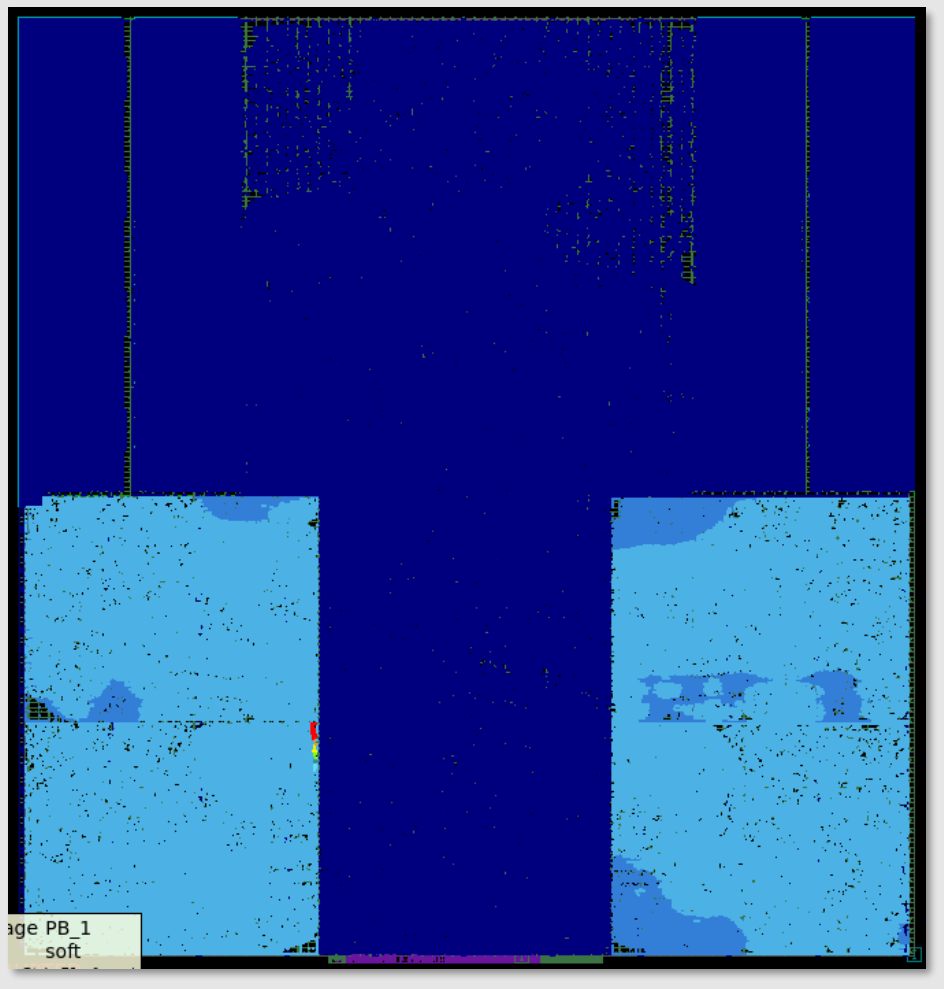


Figure 19: Static IR-Drop Map for always-on VDD and VSS (ICC2 In-Design Redhawk): Left: Weak spot in always-on supply to isolation in switchable domain “pd_core3”

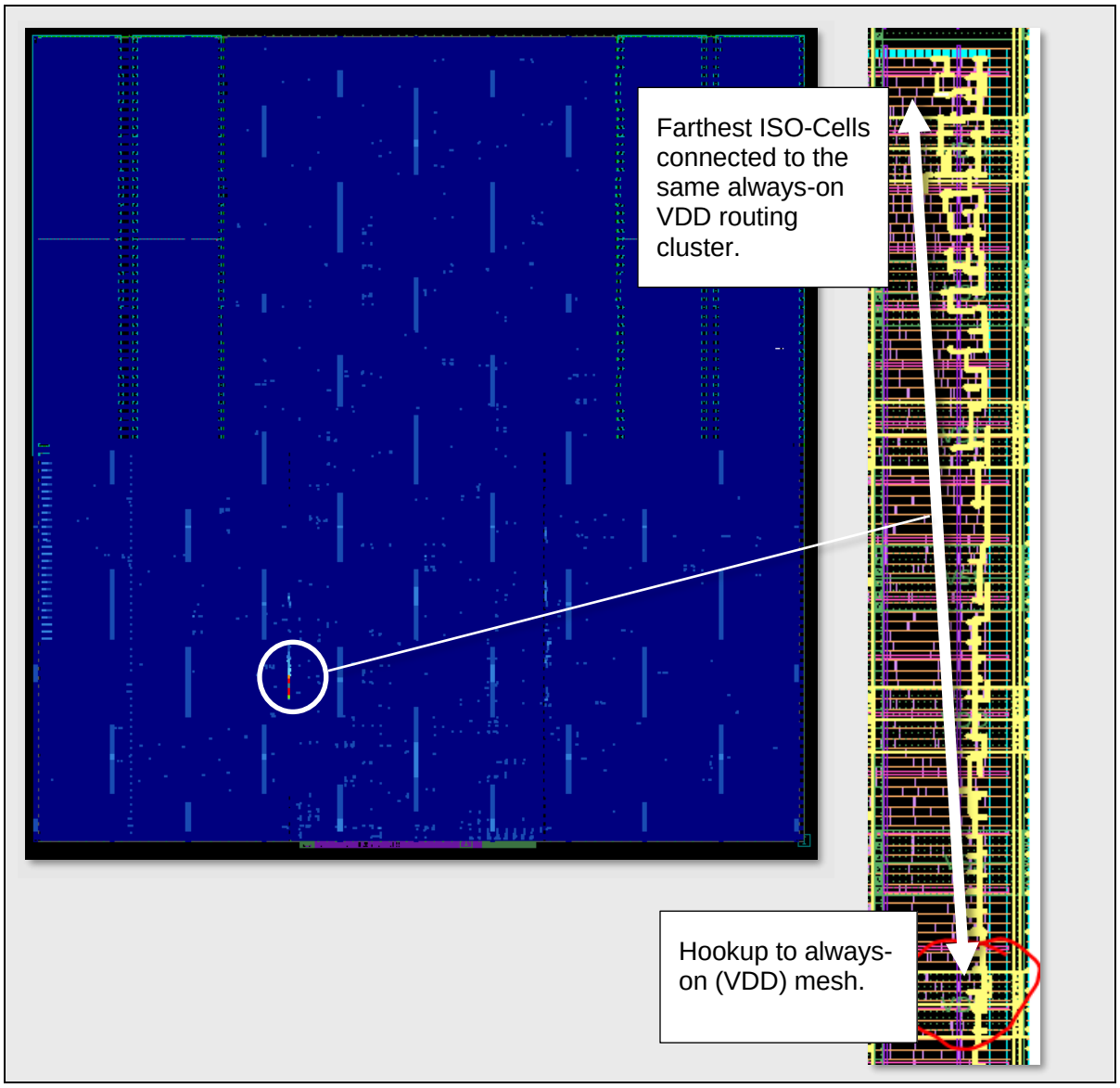


Figure 20: Electromigration Map (ICC2 In-Design Redhawk): No violations. Snippet into the weak spot.

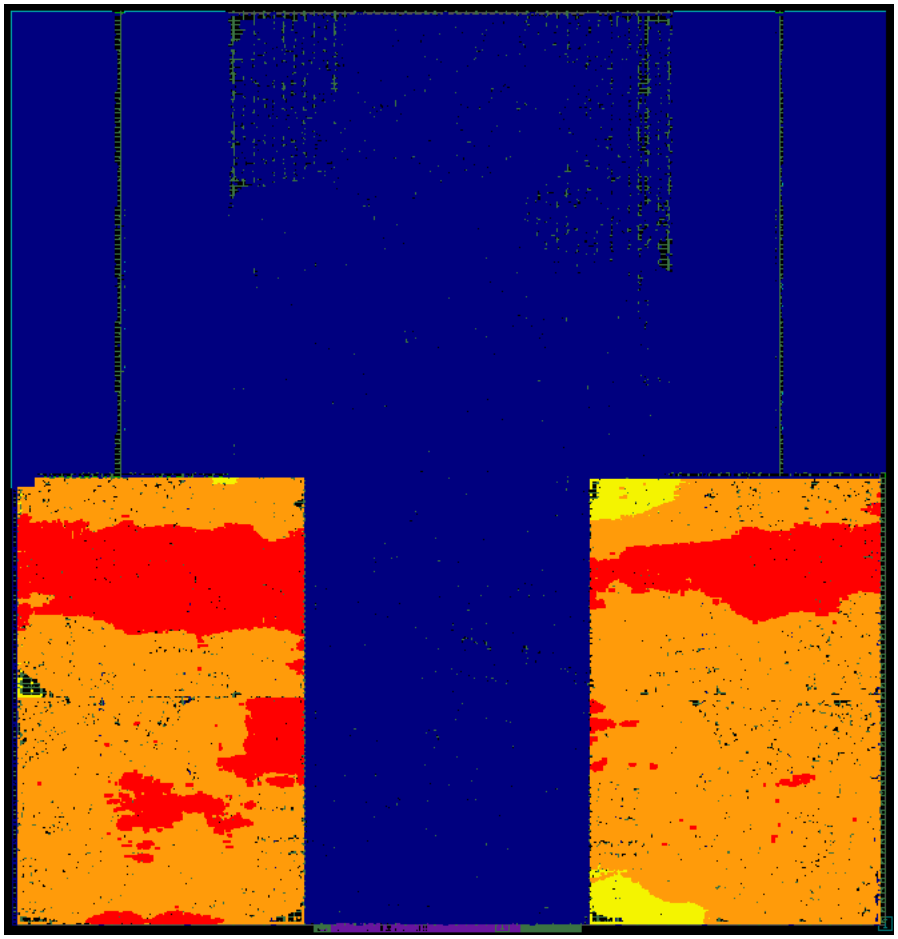


Figure 21: In-Design Redhawk Voltage Drop Map for VDD0...3 (switchable)

The maximum drop was always within the limits, with sufficient margin. The contribution of the R-On resistances of the switch cells, can be nicely seen in Figure 21 above, as an even shift in drop over all the switchable domains. The color map was chosen in a way to provide a better visual representation.

6.3 LEQ with Formality

With the presence of retention-flip-flips, additional settings are available to have Formality correctly verify RTL versus synthesized netlist if it contains retention flip-flops. This was confirmed by Synopsys.

```
set_app_var upf_infer_clamp_for_retention false
set_app_var upf_use_library_cells_for_retention true
set_app_var upf_infer_complex_retention_cells true
set_app_var upf_warn_on_unmapped_retention_cells true
```

7. Tips and Tricks

- We used `get_cells -intersect` to identify M6 straps overlapping with bias tap cells, and then, automatically shifted them by a pre-defined offset. The “-intersect” option is very powerful and available on a couple of other commands too.
- We also used `get_attribute [get_cells -hierarchical -filter \ "full_name =~ CORE_${i}__*cs_registers_i* && is_sequential==true"] \ bbox` to determine the boundary enclosing the location of all retention flip flops (via tcl “lsort” of the resulting list); then further processed that information for `create_pg_straps` commands.
- We used `get_attribute [get_site_row -intersect <coordinate of the first stripe’s start-point + small offset>] site_orientation`, to determine the orientation of the first row with retention-flip-flops (flipped, non-flipped), and thus, what’s the offset to the M3 VDDR pins respectively.
- It turned out to be beneficial to always work with a multiple of track-height and site-width, when calculating offsets, start points, etc., to allow pg-structure alignment with the std-cell library structure.
- Be aware, that certain objects provide boundary coordinates with and without margin, such as for voltage_area_shapes (“boundary” versus “bbox”), or, for macro-objects (e.g., “boundary” includes keepout margins) → important distinguishing when calculating coordinates during power planning.
- If you are stuck with pg-strategies (sometimes, they are not always intuitive to work with), use `create_pg_strap` command and do some “hand-calculation” to define the bounds of strap creation.
- Generally, utilizing the option `-voltage_areas` for various commands (such as set_pg_strategy, create_tapcell and compile_boundary_cells) is very convenient.
- `check_legality -verbose` → look out for pg_drc rules and examine the error types. This can be very helpful, in particular if certain pg_drc rules turn out to be over conservative (router techfile versus signoff DRC). Counter-measures then can be taken by utilizing the app option `place.legalize.disable_drc_rules`
- Guide bias tap cell placement in channels or, at irregular shape boundaries with placement_blockages
- Tag your individual pg-structures; this allows easy identification and follow-up actions, such as, dedicated removal of structures again; in case they have to be corrected. E.g., `create_pg_vias -nets {VDD VSS} -tag mesh_M6_lib_connect` …, allows you to remove just those vias again, if needed, by the following command: `remove_vias \ [get_vias -filter "tag == mesh_M64_lib_connect"];` this approach is also applicable to shapes, and, it can be used in conjunction with `create_pg_strap`, `compile_pg`, for instance. We also used tags to align switch cells to the vertical Always-On straps in M6, like so: `create_power_switch_array \ -lib_cell … -pg_straps [get_shapes -filter "tag==M6_mesh_VDD"] …`

8. Outlook

We have not exercised additional area reduction opportunities by the time of writing this paper. But, according to our findings, this should be possible. A comparison trial with another std-cell architecture is in the works too. Further, white space between the voltage areas is not reduced to allowed minimum yet.

A version of this paper, where technology and IP specifics are shown in detail, is going to be available on GF’s customer portal for download. Also, we are planning to transition the flow from DC-NXT/ICC2

to FusionCompiler (FC). We are already gaining quite some positive experience using FC on a different project.

As the investigation showed certain metrics like routing DRC and resulting reduction of area can be improved by using a combination of different application options. This effort takes time, and the best results could potentially be missed. Latest technologies like Synopsys' DSO.ai (Design Space Optimization AI), can exploit a broad range of potential settings to directly derive results with better QoR, like DRC, timing, and lower power.

9. Conclusions

Although, the implementation is straight forward, the "devil is in the details". Special considerations for the power mesh, dependencies to the underlying cells and how to drive the tool to optimally utilize area under the considerations of available routing resources, can make a difference, if the right tool switches are known. It turned out to be very efficient to split the issues into smaller chunks, such as, concentrating first on overall cell density optimization, and then, diving into the low power techniques, such as implementing the switchable domains, with have separate set of challenges.

10. Acknowledgements

We like to thank Michael Confal and Jens Peters from Synopsys, who helped us along our journey on tool and flow specifics. We also like to express our gratitude to ETH Zurich which provided GlobalFoundries with the RTL of the open-source PULP KRAKEN design.

11. References

[1] GlobalFoundries ABB Reference Guide (WP-000146)

[2] 22FDX Digital Guidelines (WP-000054)

[3] 22FDX-PLUS SNPS 6.75T 7.5T 9T IP based Consumer Grade Synopsys ICC2 Implementation Flow with GlobalFoundries Sign off Modules (RFL-000163)

[4] RM-scripts for DC-NXT and ICC2 (SolvNetPlus)

[5] Fusion Compiler Update Training Session

[6] SNPS Foundation IP Application Notes APN-0311 and APNT-0338

[7] DC-NXT and ICC2 man-pages

[8] Enable Level Shifter Insertion with Multiple Power and Bias Domains (SNUG World 2021)

[9] Place & Route of IoT Applications with ICC2 in Advanced Nodes, Using Low-Cost Layer Stacks (SNUG Europe 2022)

12. Appendix

12.1 Cell Density Experiments, Side by Side

The aim of sharing the below details is to make users aware of the importance of exploring the capabilities of the tools, and to regularly call for help from the Synopsys experts, who have been guiding us in this process.

App Options	Run1/Run1C	Run2	Run3	Run4
place.common.pnet_aware_density	0.4	0.4	-	-
place.common.pnet_aware_layers	M6	M6	-	-
place.coarse.auto_density_control	enhanced	enhanced	enhanced	off
place.coarse.max_density	0.48 / 0.40	-	-	0.6
place.coarse.congestion_driven_max_util	-	-	-	0.6
place.coarse.congestion_layer_aware	*Default (true)*	true	*Default (true)*	*Default (true)*
place.coarse.increased_cell_expansion	*Default (false)*	true	*Default (false)*	*Default (false)*
route.global.export_soft_congestion_maps	*Default (false)*	true	*Default (false)*	*Default (false)*
place.coarse.congestion_expansion_direction	*Default (horizontal)*	both	*Default (horizontal)*	*Default (horizontal)*
place.coarse.enable_direct_congestion_mode	*Default (false)*	true	*Default (false)*	*Default (false)*
place_opt.place.congestion_effort	*ultra (*)*	high	*ultra (*)*	*ultra (*)*
**) from icc2_pnr_setup.tcl: set SET_QOR_STRATEGY_CONGESTION_EFFOR "ultra"*				

PLACE-027 Messages

As already mentioned, looking out for the progress in the place_opt log file can provide guidance into which direction certain options could be tuned.

Run1: `Information: Automatic density control has selected the following settings: max_density 0.48, congestion_driven_max_util 0.87. (PLACE-027)`
…
`Information: Automatic density control has selected the following settings: max_density 0.48, congestion_driven_max_util 0.89. (PLACE-027)`

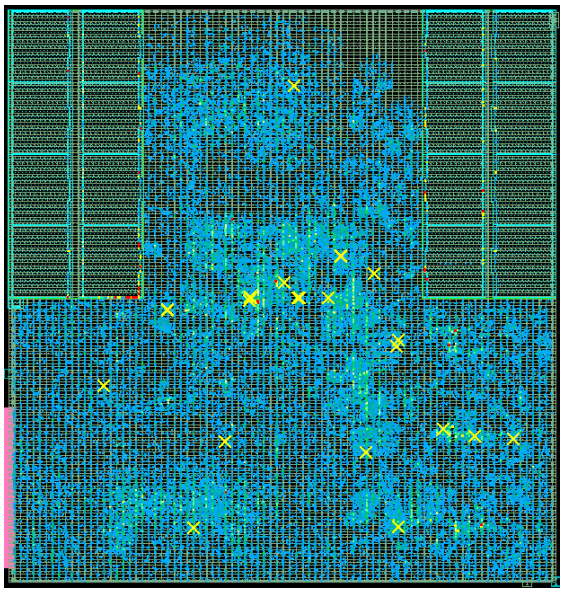
Run2: `Information: Automatic density control has selected the following settings: max_density 0.60, congestion_driven_max_util 0.87. (PLACE-027)`
…
`Information: Automatic density control has selected the following settings: max_density 0.70, congestion_driven_max_util 0.89. (PLACE-027)`

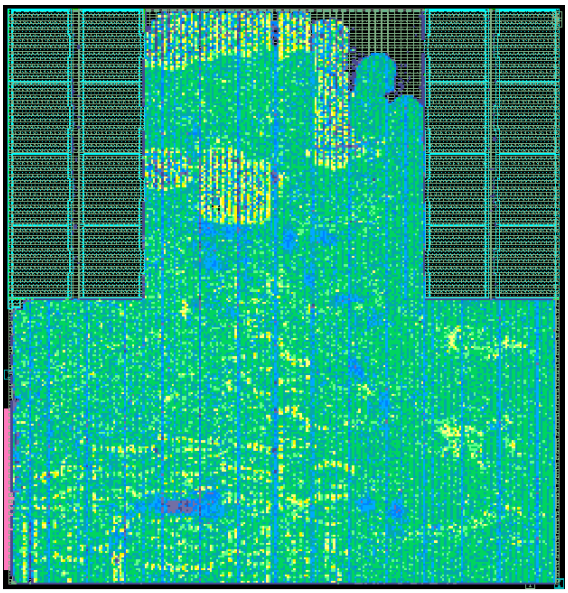
Run3: `Information: Automatic density control has selected the following settings: max_density 0.60, congestion_driven_max_util 0.87. (PLACE-027)`
…
`Information: Automatic density control has selected the following settings: max_density 0.70, congestion_driven_max_util 0.89. (PLACE-027)`

Run4: No PLACE-027 messages. This is expected since we have provided fixed value to the tool.

Run1 / Run1-C Result snippets

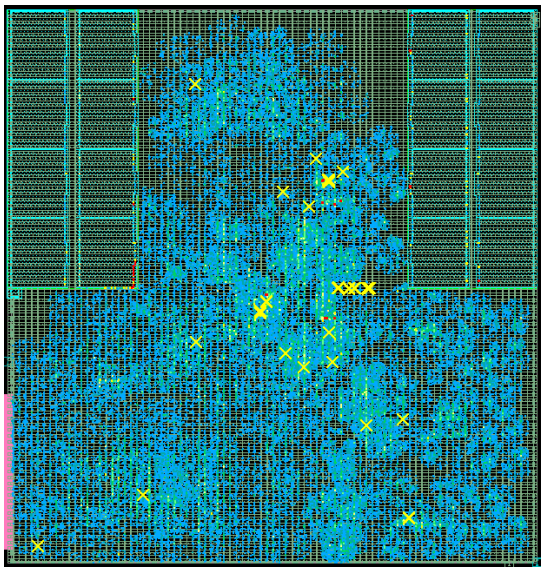
DRCs (after route_opt)	47 / 27			
QoR	(Setup)	-0.014 / -0.005	-0.014 / -0.005	1
	(Hold)	-0.000 / -0.001	-0.000 / -0.004	4 / 8
Comments	Cell distribution utilizes available space better Some room for area improvement left			

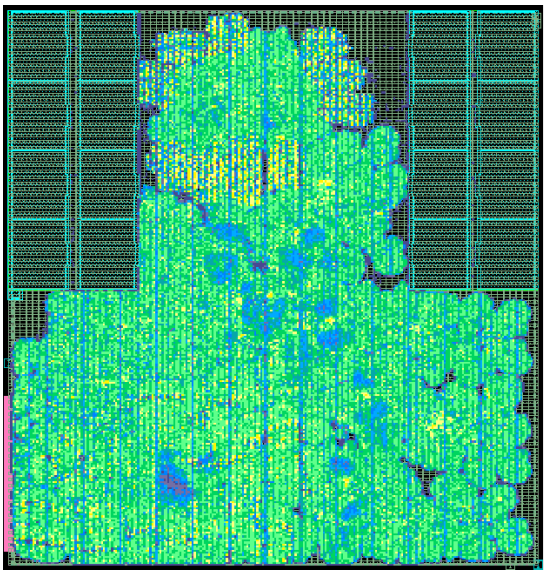




Run2 Result snippets

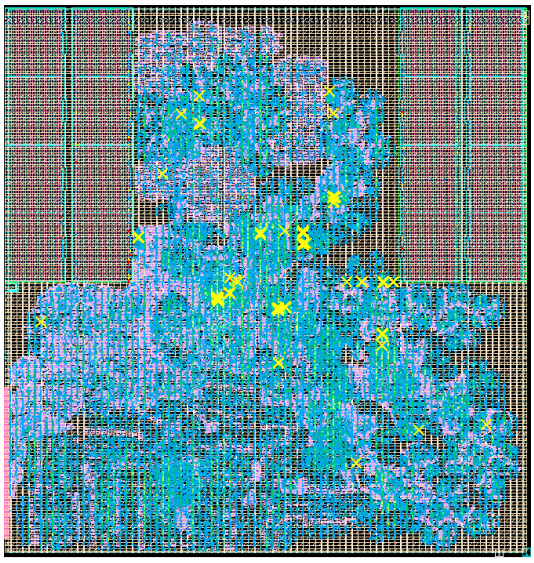
DRCs (after route_opt)	107
QoR	(Setup) 0.000 0.000 0 (Hold) -0.001 -0.002 4
Comments	Still some vertical GR overflow stripes Room for area improvement left

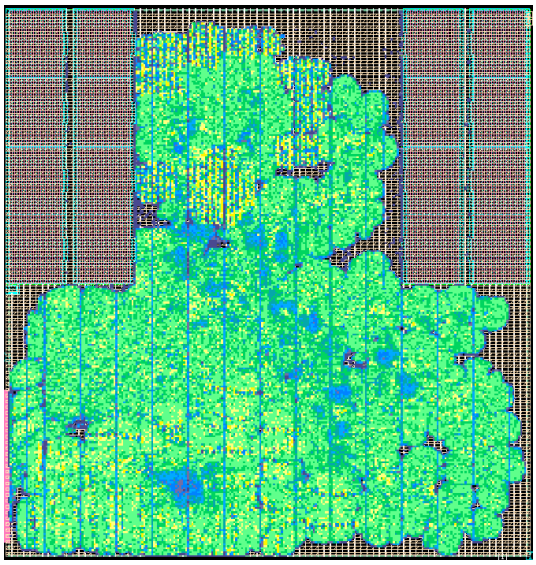




Run3 Result snippets

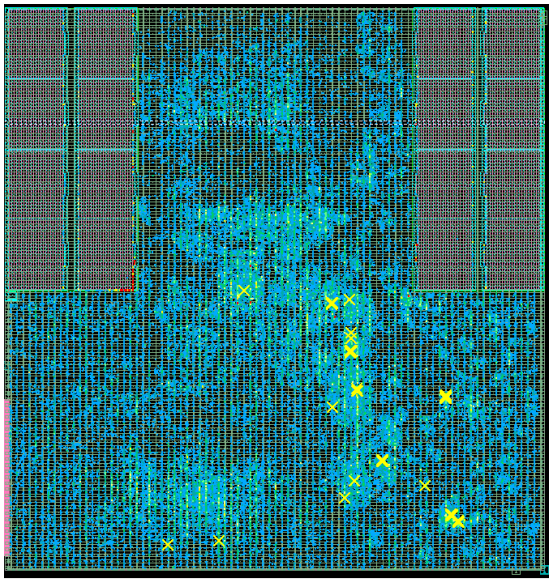
DRCs (after route_opt)	253
QoR	(Setup) 0.000 0.000 0 (Hold) -0.007 -0.014 10
Comments	auto-density control seems to utilize std-cell area better → room for further block area reduction? GR overflow under the vertical M6 straps

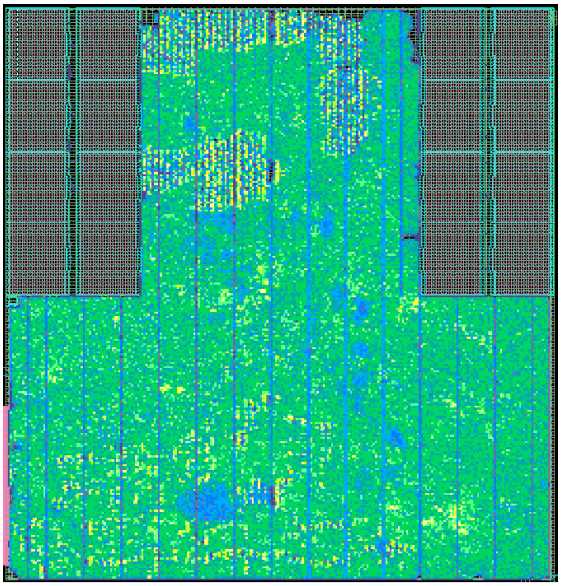




Run4 Result snippets

DRCs (after route_opt)	116
QoR	(Setup) -0.044 -0.135 6 (Hold) -0.015 -0.025 13
Comments	Better DRCs, but slightly degraded QoR. Cells utilize the available area better. Visually seems not much room for further squeezing (compared to Run3). GR overflow under the vertical M6 straps.





12.2 UPF

```
## ------------------------------------------------------------------------
## 22FDX-PLUS UPF (Version 2.1)
## ABB Corners, AO-Domain, Switchable Domain
##
## Retention-enable signals, power-down signals
## ret_en0...3:  0=disabled, 1=enabled
## pwr_off0...3: 1=disabled, 1=enabled
##
## Switching off individual cores
## CORE[0...3].core_region_i
##
## Retention of "cs_registers_i"
## Example: CORE[0...3].core_region_i/CL_CORE.RISCV_CORE/cs_registers_i
##
## SRAMs are connected to Always On (have internal switch-off)
## ------------------------------------------------------------------------

set_design_attributes -elements {.} -attribute enable_bias true
```

```
### ------------------------------------
### supply set for Always-On Domain (VDD)
### ------------------------------------

create_power_domain pd_top
create_supply_set ss_vdd
associate_supply_set ss_vdd -handle pd_top.primary

### ------------------------------------
### supply sets for Shut-Down Domains (VDD0...VDD3)
### Note: all supply sets share VSS, NET_BIAS_VNW_N and NET_BIAS_VPW_P
### ------------------------------------

create_power_domain pd_core0 -elements CORE[0].core_region_i
create_supply_set ss_vdd0 -function {ground ss_vdd.ground} \
  -function {nwell ss_vdd.nwell} -function {pwell ss_vdd.pwell}
associate_supply_set ss_vdd0 -handle pd_core0.primary

create_power_domain pd_core1 -elements CORE[1].core_region_i
create_supply_set ss_vdd1 -function {ground ss_vdd.ground} \
  -function {nwell ss_vdd.nwell} -function {pwell ss_vdd.pwell}
associate_supply_set ss_vdd1 -handle pd_core1.primary

create_power_domain pd_core2 -elements CORE[2].core_region_i
create_supply_set ss_vdd2 -function {ground ss_vdd.ground} \
  -function {nwell ss_vdd.nwell} -function {pwell ss_vdd.pwell}
associate_supply_set ss_vdd2 -handle pd_core2.primary

create_power_domain pd_core3 -elements CORE[3].core_region_i
create_supply_set ss_vdd3 -function {ground ss_vdd.ground} \
  -function {nwell ss_vdd.nwell} -function {pwell ss_vdd.pwell}
associate_supply_set ss_vdd3 -handle pd_core3.primary


##-----------------------------------------------
## Supply-Set Refinement for Always On Domain
##-----------------------------------------------

create_supply_port VDD
create_supply_net  VDD
connect_supply_net VDD -ports VDD
create_supply_set ss_vdd -function {power VDD} -update

create_supply_port VSS
create_supply_net  VSS
connect_supply_net VSS -ports VSS
create_supply_set ss_vdd -function {ground VSS} -update

create_supply_net  NET_BIAS_VNW_N
create_supply_port NET_BIAS_VNW_N
connect_supply_net NET_BIAS_VNW_N -ports NET_BIAS_VNW_N
create_supply_set ss_vdd -function {nwell NET_BIAS_VNW_N} -update

create_supply_net  NET_BIAS_VPW_P
create_supply_port NET_BIAS_VPW_P
connect_supply_net NET_BIAS_VPW_P -ports NET_BIAS_VPW_P
create_supply_set ss_vdd -function {pwell NET_BIAS_VPW_P} -update
```

```
##-------------------------------------------------
## Supply-Set Refinement for Switchable Domains
## No supply port! Will be driven by switch cells
##-------------------------------------------------

foreach i {0 1 2 3} {
  create_supply_net  VDD$i
  create_supply_set ss_vdd$i -function "power VDD$i" -update
}



### -------------------------------------
###
### Power States (limited to implementation only)
###
### -------------------------------------

add_power_state -supply ss_vdd \
  -state P0_on  {-supply_expr {power == {FULL_ON 0.59 0.72} && ground == \
  {FULL_ON 0.00} && nwell == {FULL_ON 0.00} && pwell == {FULL_ON  0.00}}} \
  -state P1_on  {-supply_expr {power == {FULL_ON 0.59 0.72} && ground == \
  {FULL_ON 0.00} && nwell == {FULL_ON [redacted]} && pwell == {FULL_ON [redacted]}}}


add_power_state -supply ss_vdd0 \
  -state P_on  {-supply_expr {power == {FULL_ON 0.59 0.72}}} \
  -state P_off {-supply_expr {power == {OFF }  }}

add_power_state -supply ss_vdd1 \
  -state P_on  {-supply_expr {power == {FULL_ON 0.59 0.72}}} \
  -state P_off {-supply_expr {power == {OFF }  }}

add_power_state -supply ss_vdd2 \
  -state P_on  {-supply_expr {power == {FULL_ON 0.59 0.72}}} \
  -state P_off {-supply_expr {power == {OFF }  }}

add_power_state -supply ss_vdd3 \
  -state P_on  {-supply_expr {power == {FULL_ON 0.59 0.72}}} \
  -state P_off {-supply_expr {power == {OFF }  }}


create_power_state_group pst_group

### for simplification, we either switch all switchable domains on, or off
### Bias is applied permanently from ABB regulator. Special considerations.

add_power_state -group pst_group \
  -state S0_all_cores_on  {-logic_expr {ss_vdd == P0_on && ss_vdd0 == P_on \
    && ss_vdd1 == P_on  && ss_vdd2 == P_on  && ss_vdd3 == P_on  }} \
  -state S0_all_cores_off {-logic_expr {ss_vdd == P1_on && ss_vdd0 == P_off \
    && ss_vdd1 == P_off && ss_vdd2 == P_off && ss_vdd3 == P_off }} \
  -state S1_all_cores_on  {-logic_expr {ss_vdd == P0_on && ss_vdd0 == P_on \
    && ss_vdd1 == P_on  && ss_vdd2 == P_on  && ss_vdd3 == P_on  }} \
  -state S1_all_cores_off {-logic_expr {ss_vdd == P1_on && ss_vdd0 == P_off \
    && ss_vdd1 == P_off && ss_vdd2 == P_off && ss_vdd3 == P_off }}
```

```
### All supplies correlate for min and max voltages (same source).
### If not done, tool might expect level shifters
set_design_attributes -elements {.} -attribute correlated_supply_group "*"


##--------------------------------
## Power Switch Section
##
## definitions for the generic switch
## (Note: using names and polarity of the actual physical switch as guidance)
## input supply  VDDP
## output supply VDDC
## enable signal EN
##
## control signal: pwr_off[0..3]
##--------------------------------

foreach i {0 1 2 3} {
  create_power_switch   SW$i \
    -domain             pd_core$i \
    -supply_set         ss_vdd$i \
    -input_supply_port  {VDDP ss_vdd.power} \
    -output_supply_port "VDDC ss_vdd$i.power" \
    -control_port       "EN pwr_off$i" \
    -on_state           {P_on VDDP {!EN}} \
    -off_state          {P_off {EN}}

   ### No mapping will happen in synthesis (only code syntax check)
   map_power_switch SW$i -domain pd_core$i -lib_cell {XXX_PGAT_XXX}
}

##--------------------------------
## Retention Section
## We only map a sub-hierarchy of each RISC-V Core to retention registers
## using the -elements option.
##--------------------------------

foreach i {0 1 2 3} {

  ## Only map the sub-hierarchy cs_registers_i
  set retention_logic \
     CORE\[$i\].core_region_i/CL_CORE.RISCV_CORE/cs_registers_i

  set_retention       my_retention_strategy_$i \
    -domain           pd_core$i \
    -retention_supply ss_vdd \
    -save_signal      "ret_en$i high" \
    -restore_signal   "ret_en$i low" \
    -elements         $retention_logic

  ### Retention-FF with Save/Restore pin, Scan and Clear
  map_retention_cell my_retention_strategy_$i -domain pd_core$i \
    -lib_cells { ... }
}
```

```
##--------------------------------
## Isolation Section
## - We only isolate outputs
## - For outputs, we will use source side isolation cells
##   (location self) to reduce clumping at VA-Boundaries
## - We exclude retention enable signal
##--------------------------------

foreach i {0 1 2 3} {

  set_isolation       my_iso_strategy_$i \
    -domain           pd_core$i \
    -isolation_supply ss_vdd \
    -isolation_signal pwr_off$i \
    -isolation_sense  low \
    -location         self \
    -applies_to       outputs \
    -clamp_value      0 \
    -exclude_elements "CORE\[$i\].core_region_i/ret_en$i
CORE\[$i\].core_region_i/pwr_off$i"

use_interface_cell my_interface_iso_strategy_$i \
 -strategy my_iso_strategy_$i -domain pd_core$i -lib_cells { … }
}

##--------------------------------
## Association of primary IOs
## (Default as below for DC-NXT and ICC2, but good practice to have it in)
##--------------------------------

set_port_attributes -attribute related_supply_default_primary TRUE \
  -elements {.}
set_port_attributes -driver_supply ss_vdd -applies_to inputs -elements {.}

##--------------------------------
## Connect SRAM pg-pins "VDD" and "VCS"
##--------------------------------

foreach ram_inst {
tcdm_banks_i/banks_gen[0].i_bank/i_macro_0/XXX
…
} {
  connect_supply_net VDD -ports $ram_inst/VDD
  connect_supply_net VDD -ports $ram_inst/VCS
}
```