UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Escola Tècnica Superior d'Enginyeria
de Telecomunicació de Barcelona

Impact of physical low power techniques in a RISC-V processor

Master Thesis
submitted to the Faculty of the
Escola Tècnica d'Enginyeria de Telecomunicació de Barcelona
Universitat Politècnica de Catalunya
by

Víctor Manuel Montabes Jiménez

In partial fulfillment
of the requirements for the master in
Electronic **ENGINEERING**

Advisor: Francesc de Borja Moll Echeto
Barcelona, Date 28-01-2021

Contents

List of Figures

Listings

List of Tables

Revision history and approval record

Revision	Date	Purpose
0	28/01/2021	Document creation
1		Document revision

DOCUMENT DISTRIBUTION LIST

Name	e-mail
Víctor Manuel Montabes Jiménez	victor.manuel.montabes@estudiant.upc.edu
Frances de Borja Moll Echeto	francesc.moll@upc.edu
Jose Antonio Rubio Sola	antonio.rubio@upc.edu

Written by:		Reviewed and approved by:	
Date	28/01/2021	Date	
Name	Víctor M.Montabes	Name	Francesc Moll
Position	Project Author	Position	Project Supervisor

Abstract

Power consumption is one of the main aspects in the overall performance of en electronic circuit. Design for low power can be applied from algorithm through the physical implementation of the integrated circuit. In this work, different low power implementation techniques will be applied to an existing System on Chip (SoC) design of a RISC-V processor. Low power approaches like multi-voltage supply, power gating and internal power optimization will be explored.

Acknowledgements

I would like to thank my family for all of this years supporting me. Also to by thesis advisor and BSC staff for giving me advice on the field.

This project is co-financed by the European Union Regional Development Fund within the framework of the ERDF Operational Program of Catalonia 2014-2020 with a grant of 50% of total cost eligible.

Generalitat de Catalunya
Departament d'Economia i Coneixement
Secretaria d'Universitats i Recerca

Unió Europea
Fons Europeu
de Desenvolupament Regional

1 Introduction

According to Moore's law, every 2 years in average the number of transistors that can be placed in a die doubles. This continuous race to make components smaller is due to the benefits that it poses to new devices. Smaller transistors decreases ASIC area and increases chip operating frequency. The ever-increasing transistor density has led to decreasing wire widths, increased current density, higher supply currents, larger transients, and large die sizes while lowering both the supply V_{min} and V_{max} of operation. Off-die dimensions, such as die-package interface, are not shrinking at the same rate. This has led to high power densities and high power consumption.

Transistor technologies below 130nm have posed a new paradigm in digital design. We can now implement tens of millions of gates on a reasonably small die, leading in a worrying power density and total power dissipation. Technology shrink has made leakage currents to increase dramatically to the point to even reach dynamic current levels. For battery-powered devices, the chip leakage is a major problem.

Today some of the most powerful microprocessor chips have an average power density of 50-75 Watts per square centimeter. This power density not only presents packaging and cooling challenges; it also can pose problems for reliability, since the mean time to failure decreases exponentially with temperature. Today, for most SoC designs, the power budget is one of the most important design goals. Exceeding the power budget can be fatal to a project, whether it means moving from a cheap plastic package to an expensive ceramic one, or causing an unacceptably poor reliability due to excessive power density, or failing to meeting the required battery life.

To combat this problem, designers are using aggressive approaches at every step of the design process, from software to architecture to implementation. These approaches include power gating, gate power optimization, multi-supply voltage and multi-threshold libraries [1].

1.1 Thesis motivation

The main motivation of this thesis is to have a deeper knowledge of all the possible alternatives to save power inside a digital design flow. Some of this low power techniques are being explored in a RISC-V single core processor.

1.2 Goals

Team specs for the used design are mainly focused in frequency and area. The operating frequency should be as close as possible to 1GHz and design area should not exceed a 1250μm x 1250μm die area, including pads, rings and hard IPs. The reason for this dimensions is that manufacturer charges the same amount for designs up to this size. With this constraints in mind, the main goal of the thesis is to probe all the mentioned low power techniques to have a knowledge basis of how much power percentage can be saved. As this chip will not be manufactured, some techniques will be approached thinking in more complex future designs.

As the original tape-out has been manufactured in TSMC 65nm technology node, another goal of this thesis will be the design migration to a GlobalFoundries 22nm FDSOI technology, ex-

ploring all the new degrees of freedom offered. Body bias use and memory macro cells migration will be key issues in this work.

Cadence tools licences available in the School have been used. Genus tool have been used for physical synthesis. Innovus tool has been used for place and route stage. Xcelium engine has been used in simulation checking. Joules engine has been used in gate-level power reporting. Voltus engine have been used for final IR-drop analysis.

1.3 Initial design overview

The Initial chip consist in a RISC-V general purpose processor capable of booting Linux jointly developed by the BSC-CNS, CIC-IPN, CNM and UPC [3]. The core design is based on the Lagarto chip, designed by the CIC-IPN with educational purposes. As the current design will be the basis of future high-complexity projects like DRAC project, the chip has been named preDRAC.

Figure 1 shows the block diagram of the design. It incorporates a 5-stage pipeline that implements the 64-bit RV64IMA scalar ISA, as well as the associated instruction and data caches, a unified L2 cache and the peripherals required to connect the processor with external devices. Finally, table 1 summarizes the main IP blocks. Some blocks in the design should be deeply explained.

The core unit architecture is based in a 5-stage pipeline. Every stage in the process has is own sub-module in charge (FETCH, DECODE, READ_REGISTER, EXECUTION, WRITE BACK). In addition, some units need memory macros to operate. The ”FETCH” unit has 6 SRAM macro memories in a Bimodal Predictor. The ”READ_REGISTER” unit has also 2 macro register files associated. The core unit is directly connected to all the first level cache memories, the Control/Status Register (CSR) and the Performance Monitoring Unit (PMU).

The cache hierarchy is composed in 2 levels. Level 1 cache (L1) memories are tightly connected to core unit and are the first computation support. They are divided in data caches and instruction caches. 4 macro register files are used in each cache block. Level 2 (L2) cache memory is used to have an active copy of all level 1 caches. The L2 is divided in different modules. There are the memory modules it selves, the data module, which contains 1 big SRAM macro memory, and the metadata module, which contains 2 macro register files. Also, there are the agent modules. In this case there are the Acquire Tracker, the Voluntary Release Tracker and the Write Back Unit. The chosen mechanism to communicate between the different levels of cache as well as to the memory controller is TileLink.

Some peripherals are needed to communicate the ASIC with other devices. In this design, an UART unit is used to transmit and receive serial data from the host PC to the PCB. The UART module is composed mainly of 4 submodules: the UART core unit, 1 transmitter, 1 receiver and a UART controller. There is also an SD card controller which is in charge of reading the Linux OS image from the attached SD card through a series of AXI-lite read/write transactions and SPI protocol packets.

A Performance Monitoring Unit (PMU) is used to to be able to measure the events that happen inside the processor in order to understand the effects future applications over the micro-

architecture . As the chip implementation focuses on scalability and flexibility, the PMU can be configured at any time by the processor.

Since a bus connection between ASIC and FPGA was required, a FMC cable was used. Through this FMC cable, a packetizer which that breaks down AXI transactions into packets and sends them through a narrower link has been implemented. Additionally, a JTAG communication interface between PC and in-out FIFOs has been added to the ASIC.

Finally, a Debug Ring module has been inserted to verify correct understanding between simulation and FPGA wrappers. Table 1 summarizes ASIC modules description.

Table 1: IP blocks summary

IP block	Description
Core	RV64IMA 5-stage in-order pipeline
Instruction cache	4 4-way 16KB cache blocks
Data cache	4 4-way 16KB cache blocks
L2 cache	1 8-way 64KB cache block
TileLink	128-bit wide
UART	AXI4-Lite with slave interface
SD card controller	AXI4-Lite with slave interface. SPI packets.
JTAG	PC interface with internal FIFO
Packetizer	XX bits
PMU	Performance Management. User accesible.
Debug Ring	write lo L2 cache

1.4 GLOBALFOUNDRIES 22FDX technology

The technology used in this work is GLOBALFOUNDRIES 22FDX. Technical information of technology has been extracted from [5]. This consist in 22nm Fully-depleted Silicon on Insulator (FDSOI). Silicon-on-insulator technology refers to the use of a layered SOI substrate instead of a conventional bulk substrate. Fully-depleted relies on a ultra-thin layer of an insulator, called buried oxide, placed on top of the base silicon. Figure 2 shows a schematic view of the 2 technologies.

The main advantage of the use of this technology versus bulk transistors is that leakage currents are decreased thanks to the layer. Another advantage is that the variability is smaller across die due to a lower doping effort. The main disadvantage is the cost. SOI substrates are more expensive than bulk CMOS wafers.

One of the main aspects of this technology is the viability of body biasing. Body bias offers an additional option to tune cell performance of power. The early conditions PVT (process, Voltage, Temperature) now can be added PVTB. Body bias consists in tuning the bias voltage. Depending on the delta of V_{bs}, a body bias voltage can increase or decrease threshold voltage of the device There are 2 modes of body biasing:

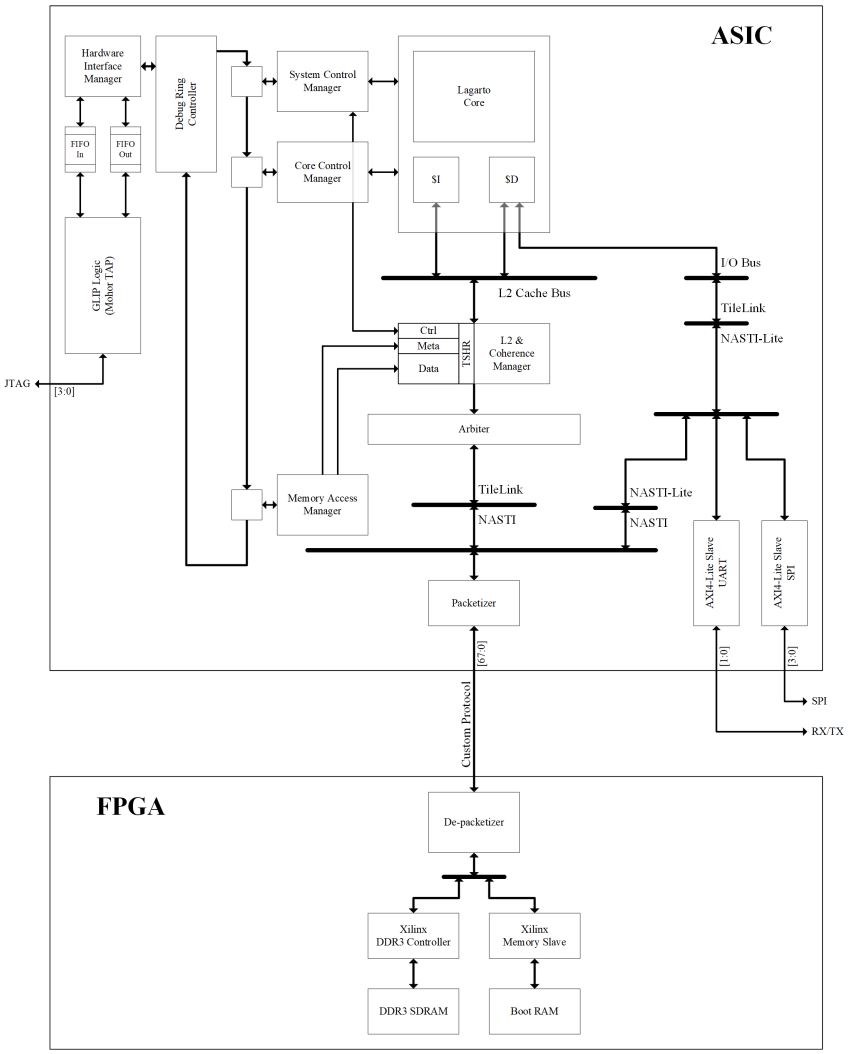


Figure 1: pre-DRAC processor block diagram

- Forward body biasing: consists in having a positive V_{bs}. This technique decreases threshold voltage improving cells performance but increasing the leakage. In this technology, forward body bias is performed by applying a positive voltage in the NMOS substrate and a negative voltage in the PMOS substrate. The voltages must be symmetrical but with different signs.
- Reverse body biasing: consists in having a negative V_{bs}. This technique raises threshold voltage lowering the device leakage but increasing also delays and thus, decreasing cells performance. In this technology, forward body bias is performed by applying a positive voltage in the PMOS substrate and a negative voltage in the NMOS substrate.

In this work, the chip must operate at the highest frequency possible due to the high budget in mind. The new design (DRAC in order) should reach 1GHz, so our work in this thesis is to get as close as possible to that frequency while minimizing the power. For that reason, reverse body bias is discarded of our work and we developed a chip only with FBB option.

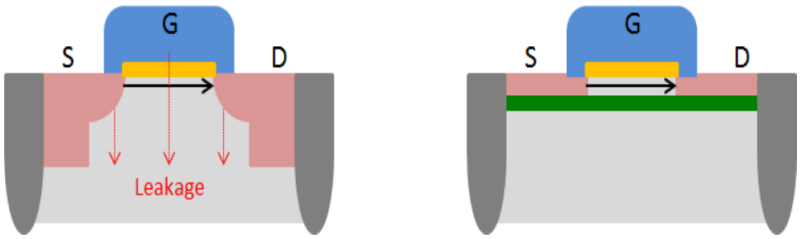


Figure 2: Bulk scheme (left) vs FDSOI scheme (right). The SOI layer prevent leakage currents to flow across the device.

This technology offers a wide range of available libraries [13]. 3 different supply voltages are offered for all the cells: 0.9V, 0.8V and 0.65V. Moreover, all the libraries are characterized in 3 process corners (SSG, FFG, and TT) and 4 different voltages (-40^{o}C, 25^{o}C, 85^{o}C, 125^{o}C). There is also a distinction on V_{th}. Libraries are divided in high V_{th}, regular V_{th}, low V_{th} and super low V_{th}. In this work, as we are looking first to the maximum ASIC performance, low and super low V_{th} libraries have been used in the "TT" corner.

A low power kit (LPK) is also included in libraries with all the required cells to perform low power techniques: isolation cells, high-to-low level shifters, bidirectional level shifters, power switches, always-on cells and retention flops.

Finally, memory compilers are available to characterize single and dual port SRAMs, single and dual port register files and ROM memories.

1.5 Unified Power Format (UPF)

Nowadays, low power designs are designed together with HDL and power description files. The two main formats used for power description are the Common Power Format (CPF) developed by Cadence and the Unified Power Format (UPF) developed by Synopsys. The last one has become in the last years the industry standard and, consequently, the most used option [2].

The Unified Power Format provides the ability for electronic systems to be designed with power as a key consideration early in the process. UPF provides a consistent format to specify power-design information that may not be easily specifiable in a hardware description language or when it is undesirable to specify the power semantics in an HDL, as doing so would tie the logic specification directly to a constrained power implementation. Modern synthesis tools are optimized with the format so as to it can be inserted and updated in the flow, creating a power intent specification in parallel with the logical intent. Every step of a digital design process creates a new UPF file updated with the chip specs which will be used in the next step. Figure 3 shows UPF supporting the entire design flow.

The UPF format is able to describe some source files. This collection of source files is the input to several tools, e.g., simulation tools, synthesis tools, and formal verification tools. UPF supports the successive refinement methodology where power intent information grows along the design flow to provide needed information for each design stage.

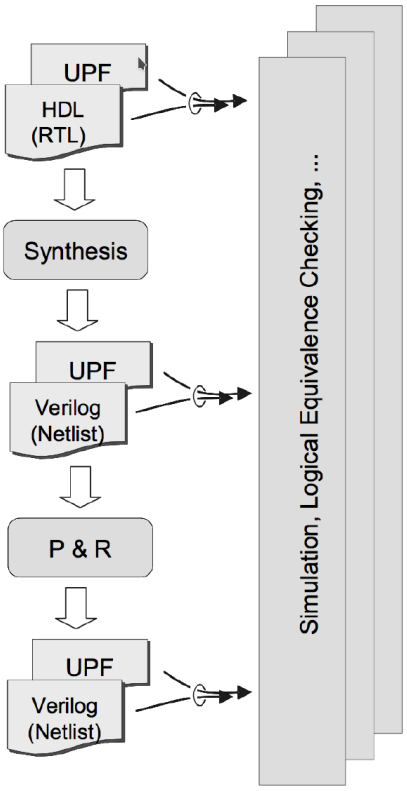


Figure 3: UPF tool flow.

- Simulation tools can read the HDL/UPF design input files and perform RTL power-aware simulation. At this stage, the UPF might only contain abstract models such as power domains and supply sets.
- In synthesis tools the UPF may be refined to add implementation-related information. His further-refined specification may then be processed to produce a netlist.
- Place and route tools read both the netlist and the UPF files. In this stage, additional items like power switches might be added.

In this work we have used the Unified power Format due to the wide variety of information sources and to follow industry tendencies. To avoid possible format problems with modern synthesis tools (we use 2019 versions), the version 3.0 of the power format have been chosen. A UPF file consist in the next points:

- Define bias ports and nets.
- Connect ports and nets.
- Create supply sets, that will link power domains and supply ports connection.
- Create power domains, which are the voltage islands of our design and will define also the power gated modules.
- Define the different states of power domains.

- Define level shifting, isolation, retention and power switching strategies.

This steps will be deeply explained in the next chapters.

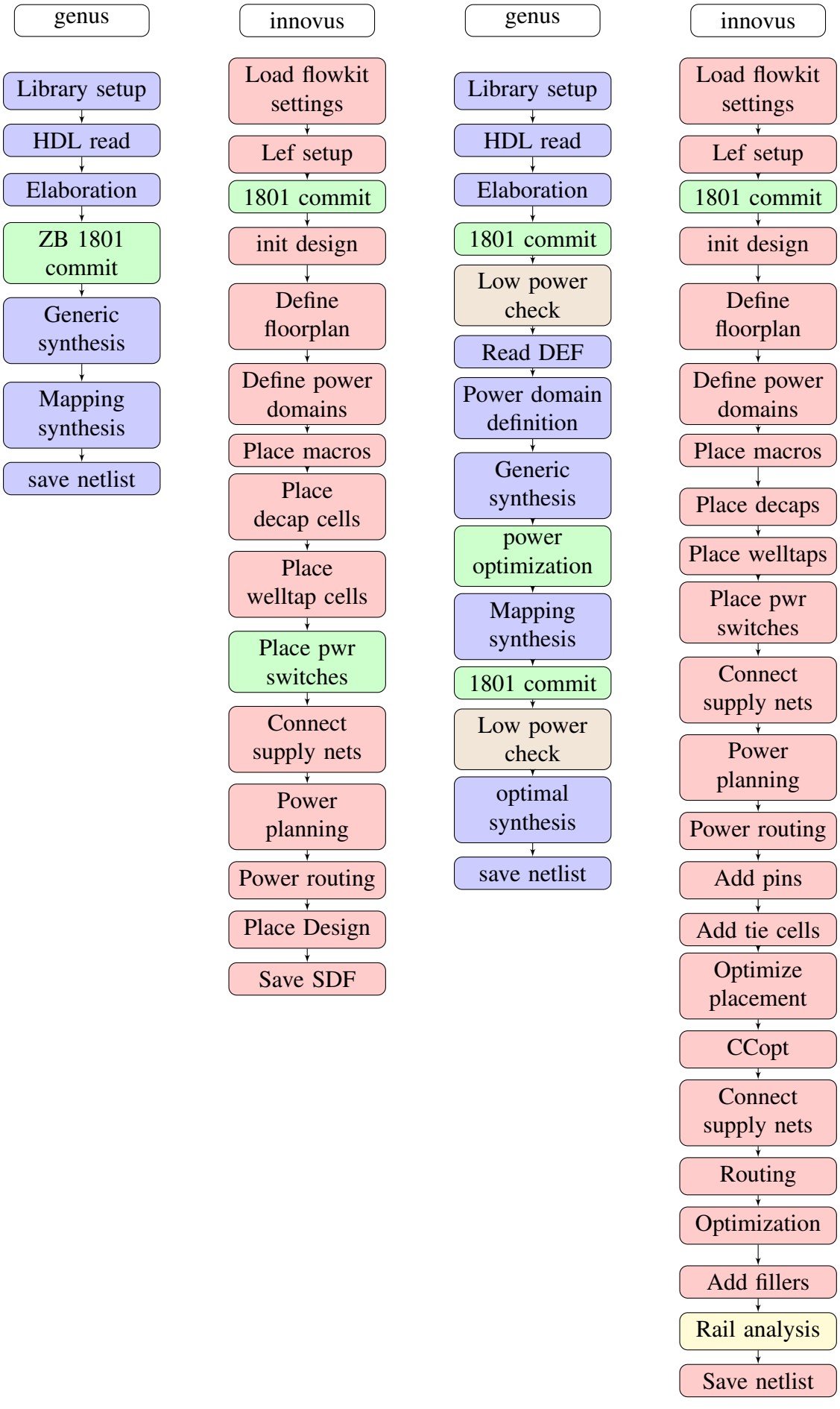
2 Digital design flow

2.1 General flow

The project flow structure is based in a low power Rapid Adoption Kit developed by Cadence Support [4] and the respective synthesis and P&R user guides [12] and [11]. It consists in a 4-step single corner flow that contains a preliminary phase and a final phase:

- First, a simple synthesis is performed from the original RTL code. Only the default voltage and libraries are used in this step. Thus, from a point of view of the original flow, no power intent should be used. However, GF-FDSOI cells need power intent info for body biasing even if it is zero bias, so a minimal power intent file with power and ground connections have been used. The main purpose of this step is get a minimal netlist for the next step.
- Secondly, the initial P&R is performed. A primitive Floorplan is elaborated with multi-domain power rings and UPF power intent info. The purpose if this step is to get information of the std cells and macros future position as well as to detect possible area or design problems in an early phase. At the end of the step, a ”def” output file will be used in the third step.
- Thirdly, the physical synthesis is performed. In this, step, the RTL is re-elabolated with all the power intent info (multi-domain, power switching, body biasing etc.). The def file is loaded before synthesis to get an idea of design distribution. In this stage, some low power intent cells are inserted into the netlist. Specifically, level shifters and isolation cells are inserted. In references, the authors advise to do it in P&R stage, but in this version of Synthesis tool, level shifter insertion is better optimized than in the P&R tool, so it is better in this case to do it early. Conformal engine is used after every power intent commit to check possible problems in physical cells insertion.
- Finally, P&R is performed. The design distribution is identical to the pre-P&R stage, but in this design the physical netlist is used. Power switches, welltap cells and decap cells are inserted in this phase. At the end of the flow, Voltus tool is invoked to perform a static rail analysis.

In the scheme, Genus commands are colored blue, Innovus commands are colored red, Conformal commands are colored brown and Voltus commands are colored yellow. The green commands are the ones directly related with low power insertion.


genus
Library setup
HDL read
Elaboration
ZB 1801 commit
Generic synthesis
Mapping synthesis
save netlist
innovus
Load flowkit settings
Lef setup
1801 commit
init design
Define floorplan
Define power domains
Place macros
Place decap cells
Place welltap cells
Place pwr switches
Connect supply nets
Power planning
Power routing
Place Design
Save SDF
genus
Library setup
HDL read
Elaboration
1801 commit
Low power check
Read DEF
Power domain definition
Generic synthesis
power optimization
Mapping synthesis
1801 commit
Low power check
optimal synthesis
save netlist
innovus
Load flowkit settings
Lef setup
1801 commit
init design
Define floorplan
Define power domains
Place macros
Place decaps
Place welltaps
Place pwr switches
Connect supply nets
Power planning
Power routing
Add pins
Add tie cells
Optimize placement
CCopt
Connect supply nets
Routing
Optimization
Add fillers
Rail analysis
Save netlist

2.2 Logic simulation

After chip design, Xcelium tool have been used to check logical concurrence. The graphical user interface Simvision has been used following the user manual [10].

Xcelium lets the user to get gate level simulations, this is, timing simulations with realistic gate delays. In order to simulate accordingly, both the netlist and a a gate delay file (SDF) are needed from chip output files. The tool commands needed to run the code compilation and simulation are the following:

```
xrun -clean
xrun -f <name_of_run_file>.f
xrun -R -input <name_of_simulation_file>.tcl -gui
```

As it will be explained in power gating chapter, new verilog modules have been designed for power shut off purposes. Because of that, this modules have been needed to be tested in all the flow phases. Post-P&R and post-synthesis netlists have been probed in Xcelium as well as source RTL.

The simulation profiles used for testing are the ISA tests available at RISC-V repository in GitHub. For low power simulation, the most appropriate tests are the machine level ones. That is, the ones with the highest level of permisons to perform instructions. The reasons for choosing that tests is that they are the quicker and closer to gate level understanding. Table 2 shows a summary of the tests used.

Table 2: ISA tests summary

Test	Description
rv64mi-p-csr	Read-modify-write Control Status Registers in design
rv64mi-p-illegal	Predefined illegal instruction routine
rv64mi-p-ma-addr	Address exchange routine
rv64mi-p-ma-fetch	Fetch core instruction routine
rv64mi-p-sbreak	Instruction used to return control to a debugging environment
rv64mi-p-scall	Instruction used to make a service request to the execution environment
rv64mi-p-timer	Read the full 64 bits of the cycle, time, and instret counters
hello	A ”Hello World” message is written into the UART

After checking the correct behaviour of the waveforms, net stimulus files are needed for gate level power simulation. Both SAIF and TCF formats are adecuate to save the info. In this work, net toggle information have been saved in SAIF files because there was previous experience in the work group with this format. XCelium commands are used after simulation to save test activity:

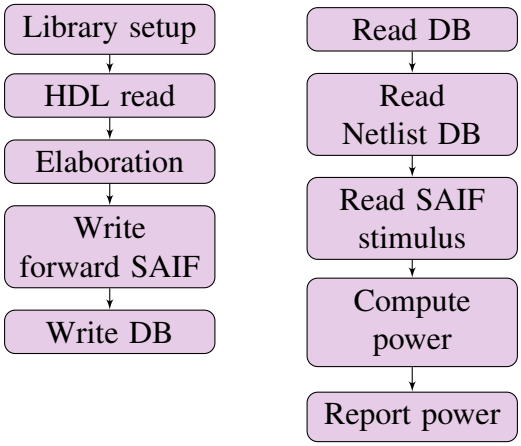
```
dumpsaif -scope tb.DUT.Top_asic_inst -depth to_cells -internal -memories \
    -output <file_name>.saif
```

2.3 Power evaluation

The last step in the work has been Joules power reporting. Gate level reporting must be done in 2 steps [9]:

- Joules pre-step: the original RTL is elaborated and saved in a forward SAIF.
- Joules power reporting: stimulus and forward SAIF are computed with the final design netlist. Then gate-level power is reported in every simulation case.

The last scheme shows a summary of the 2 Joules steps, colored in violet:



3 Project development

The power reduction steps have been done following an order. Firstly, the most invasive techniques which needed deeper changes in chip distribution like multi-voltage supply and power shut off have been done. Internal power optimization has been approached after all the steps were done to have a clear variable in mind for maximum power values.

3.1 Initial design synthesis and floorplan

The first design migration have been focused on fulfill the proposed specs. The supply voltage and body biasing level are the highest as well as the synthesis effort. With this, the design has a supply voltage of 0.9V, a nwell voltage of 0.9V and a pwell voltage of -0.9V.

In these conditions, the maximum achieved frequency with the given RTL is 625MHz. This will be the operating frequency that we will carry in all the power reduction steps.

Regarding the power specification, in spite of being a single voltage design, the body biasing requires the definition of a UPF file with the forward body bias information [7]. This file has been defined as follows:

- Supply and body bias ports and nets have been defined and linked:

```
create_supply_port VCH
create_supply_port VSS
create_supply_port VCNW
create_supply_port VCPW

create_supply_net VCH
create_supply_net VSS
create_supply_net VCNW
create_supply_net VCPW

connect_supply_net VCH -ports VCH
connect_supply_net VSS -ports VSS
connect_supply_net VCNW -ports VCNW
connect_supply_net VCPW -ports VCPW
```

- Supply sets are defined. 2 dummy supply sets are made with the body bias ports to join voltages in the view definition file [8].

```
create_supply_set ss_aon_v09_bb         -function {power VCH}  -function {nwell
     VCNW} -function {pwell VCPW} -function {ground VSS}
create_supply_set ss_VNW                -function {power VCNW} -function {
    ground VSS}
create_supply_set ss_VPW                -function {power VCPW} -function {
    ground VSS}
```

- Power domains are defined. 2 dummy power domains are used with the same purpose as the supply sets.

```
create_power_domain AO_V09_BB            -elements {.}        -supply {primary
    ss_aon_v09_bb}
create_power_domain DUMMY_PD_VNW         -elements {} -supply {primary ss_VNW}
create_power_domain DUMMY_PD_VPW         -elements {} -supply {primary ss_VPW}
```

- Finally, power states define the operating conditions of the power domain. Dummy power domains do not need this condition.

```
add_power_state -supply AO_V09_BB.primary -state { h     -supply_expr { power
    == {FULL_ON 0.9}  &&  ground == {FULL_ON 0.0} &&  nwell == {FULL_ON 0.9} &&
      pwell == {FULL_ON -0.9}} -simstate NORMAL }
add_power_state AO_V09_BB  -state { PM1  -logic_expr {AO_V09_BB.primary == h }}
```

In the view definition file, dummy power domains are getting their operating conditions as well as the power bias:

```
create_op_cond -name AO_V09_BB_TT_0P90V_0P00V_0P00V_0P00V_25C -V {0.9} -T {25.0} -P
    {1} -library_file {...
create_op_cond -name DUMMY_VNW_oc -V {0.9} -T {25.0} -P {1} -library_file {...
create_op_cond -name DUMMY_VPW_oc -V {-0.9} -T {25.0} -P {1} -library_file {...
...
update_delay_corner -name _default_delay_corner_ -power_domain AO_V08_BB...
update_delay_corner -name _default_delay_corner_ -power_domain DUMMY_PD_VNW -opcond
    DUMMY_VNW_oc...
update_delay_corner -name _default_delay_corner_ -power_domain DUMMY_PD_VPW -opcond
    DUMMY_VPW_oc...
```

Physical synthesis results show a design area of 0.414 mm^2, a power consumption of 199.84 mW. Timing results show a capable of work design, but with 40 paths below 10 ps in slack (all of them in the core unit and cache memories). This gives us an idea of the low margin we have to implement techniques that can affect timing.

Respect to the P&R stage, Figure 4 shows Top view of the floorplan with and without routing. The die area has been fixed in a 850μm x 850μm square while the area assigned to pads is the rest of the perimeter with 150 μm width. This is a total floorplan area of 1150μm x 1150μm, which fulfills team specs. As can been seen in figure, all the components fit in the die without any problem. In fact, the design could be even smaller, but as the manufacturing cost would be similar, we will keep it as is. Power pads are not inserted in this design.

A total amount of 4 power rings have been needed to power all the ASIC. Table 3 shows a summary of the used rings and their metal layer.

Table 3: Initial power rings summary

Ring	Voltage (V)	Use	layer
VCC	0.9	Main supply	JA JB (metal 9 and 10)
VSS	0	Ground	JA JB (metal 9 and 10)
VCNW	0.9	N bias	C1 (metal 3)
VCPW	-0.9	P bias	C1 (metal 3)

To make power planning as easy as possible, memory macros have been ordered to get rectangular macro areas. Moreover, all the macros are faced to the center of the chip to facilitate routing. Table 4 shows a summary of the macro memory cells used and their size.

Power and body bias rings and stripes have different widths. Body bias power planning is considerably thinner than power bias. This is because body bias rings do not drive current to cells. Body bias mechanism is controlled only by voltage difference. Then, it is no need to fulfill IR drop conditions in this power rings. Figure 4 shows a zoom comparison of the different rings.

In addition to power rings, welltap cells and boundary cells are required in this technology. Boundary cells are used in domain and die limits to create a safe enclosure protecting boundary

Table 4: Macro memory cells summary

Block	Type	size	number
L2 data cache	SRAM	4096x128	1
L1 instruction cache	RF	256x128	4
L1 data cache	RF	256x128	4
Bimodal Predictor BTB	SRAM	1024x40	2
Bimodal Predictor BIPC	SRAM	1024x28	2
Bimodal Predictor PHT	SRAM	1024x4	2
INT_REGISTER file	RF	32x64	2
L2 cache meta tag A	RF	128x128	1
L2 cache meta tag B	RF	128x48	1
L2 cache meta tag 1	RF	64x88	1
IC tag array	RF	64x80	1

cells gates. Every limit (upper, lower, left, right and corners) have a different set of boundary cells assigned for this technology. The next code shows the command used to define them:

```
setEndCapMode -topEdge SC8T_COLCAPPX1_CSC20L
              -bottomEdge SC8T_COLCAPNX1_CSC20L \
              -leftBottomCorner SC8T_CNREXTANTENNANRX11_CSC20L \
              -leftEdge SC8T_ROWCAPANTENNARX11_CSC20L \
              -leftTopCorner SC8T_CNREXTANTENNAPRX11_CSC20L \
              -rightBottomCorner SC8T_CNREXTANTENNANLX11_CSC20L \
              -rightBottomEdge SC8T_CONCAVENLX11_CSC20L \
              -rightEdge SC8T_ROWCAPANTENNALX11_CSC20L \
              -rightTopCorner SC8T_CNREXTANTENNAPLX11_CSC20L \
              -rightTopEdge SC8T_CONCAVEPLX11_CSC20L \
              -leftBottomEdge SC8T_CONCAVENRX11_CSC20L \
              -leftTopEdge SC8T_CONCAVEPRX11_CSC20L
addEndCap
```

Welltap cells are physical cells used to connect the appropriate bias to nwell and pwell. They contain the well ties that is usually removed from standard cells for area reduction. Welltap cell power planning has special commands that should be pointed:

- First, welltap cells need to be declared and placed. The distance between cells should not exceed 80μm to avoid possible DRC problems.

```
addWellTap -cell SC8T_TAPX8_CSC20L -cellInterval 70 -prefix WELLTAP_09
```

- Secondly, power stripes are placed manually crossing the welltap cells contacts.

```
addStripe -nets {VCNW VCPW} -layer C1 -direction vertical -width 0.044 -spacing
     0.352 -start_offset 80.166 -number_of_sets 1
    -block_ring_top_layer_limit...
...
```

- Finally, connections are specified and stripe bias are declared.

```
globalNetConnect VCNW -inst *WELLTAP_09_* -override -pin VNW_N -type pgpin
globalNetConnect VCPW -inst *WELLTAP_09_* -override -pin VPW_P -type pgpin
setViaGenMode...
editPowerVia -add_vias 1 -bottom_layer M2 -nets VCNW -skip_via_on_pin {}
    -split_vias 1 -top_layer C1
editPowerVia -add_vias 1 -bottom_layer M2 -nets VCPW -skip_via_on_pin {}
    -split_vias 1 -top_layer C1
```

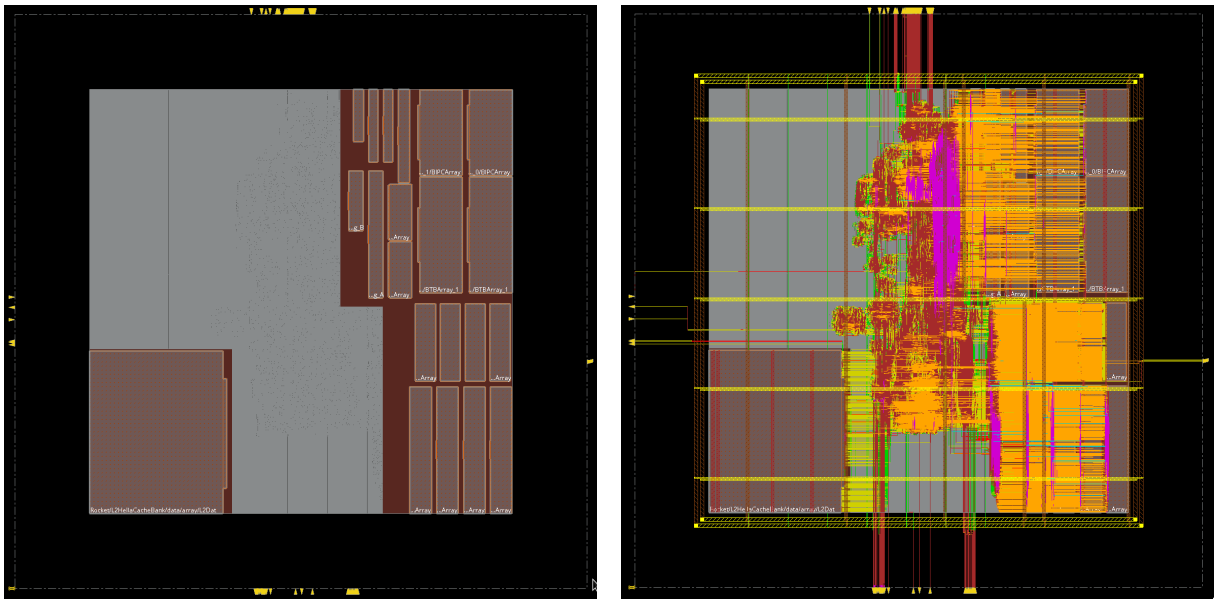


Figure 4: Floorplan without routing on the left and with routing on the right.

Figure 5 shows the power planning of some welltap cells. As can be seen in Figure 4. Standard cell area is clearly under the chip size. The post-syn area result is 0.399 mm^2. The power consumption is summarized in table 5.

Table 5: post-pnr default model power prediction

	Power (mW)	Percentage (%)
Total dynamic power	156.33	84.94
Total leakage power	27.71	15.06
Total power	184.05	100

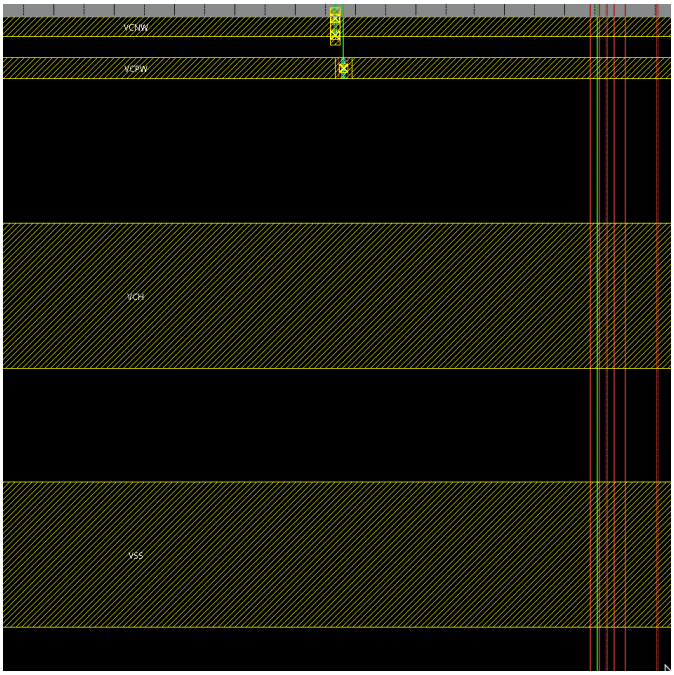


Figure 5: Power and body bias rings comparison.

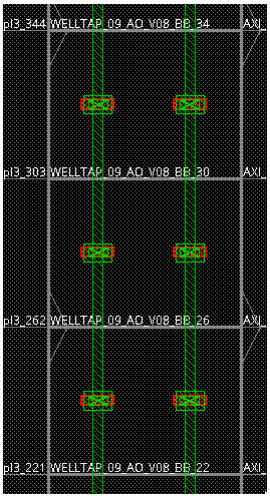


Figure 6: Welltap routing.

3.2 Supply and Body biasing comparison

The first step to explore multi-voltage supply is to compare the different alternatives of supply and body bias voltage we have. For that purpose, we have simulated the chip response with different combinations of bias voltage and bias voltage. 6 different approaches have been taken into account:

- 0.9V supply voltage with zero bias.
- 0.9V supply voltage with maximum bias (nwell 0.9V, pwell -0.9V).
- 0.8V supply voltage with zero bias.
- 0.8V supply voltage with maximum bias (nwell 0.8V, pwell -0.8V).
- 0.65V supply voltage with zero bias.
- 0.65V supply voltage with maximum bias (nwell 0.8V, pwell -0.8V).

The goal of this comparative is to look for the difference in voltage and the maximum potential of forward body biasing. In summary, we want to balance whether zero bias high voltage options are preferable to forward bias low voltage options (eg: 0.9V zero bias vs 0.8V forward bias). The power and area results of the comparative can be seen in table 6.In terms of timing, the worst 20 paths of each design can be seen in Figure 7.

Table 6: Body bias power comparison. Post-synthesis results measured at 500MHz.

	Power (mW)	Area (mm^2)
0.9V FB	170.63	0.414
0.9V ZB	144.1	0.415
0.8V FB	132.97	0.415
0.8V ZB	118.1	0.416
0.65V FB	93.25	0.418
0.65V ZB	81.32	0.421

As can be seen, as this design is not highly area-constrained, the differences of different supplies in terms of area are minimal. In terms of timing, there is a small advantage of choosing lower voltage forward bias solutions instead of zero bias ones, except for 0.9V. For example, both 0.65V FB and 0.8FB solutions have similar positive slack results. It is important to remark that all the models except the zero bias at 0.65V fulfill timing specs. In terms of power, there is a clear advantage in power reduction by choosing forward bias options instead of zero bias ones. As a conclusion, in this work forward bias power domains are the most suitable solution and will be the ones chosen for the rest of the work.

3.3 Applying Multi-voltage design

The first step in power efficiency is to look for static voltage reduction opportunities, which is called Static Voltage Scaling. The most basic form of this new approach is to partition the internal logic of the chip into multiple voltage regions or power domains, each with its own supply.

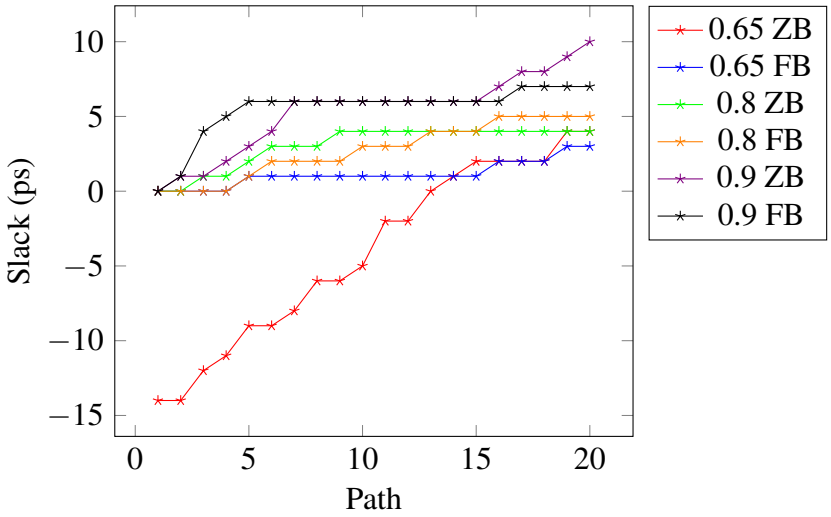


Figure 7: 20 worst slack comparative of each supply alternative.

It is based on the realization that in a modern SoC design, different blocks have different performance objectives and constraints [1]. The main purpose of this technique is to reduce dynamic power, which is quadratically dependent of bias voltage. This means that a 0.25V voltage decrease could supone up to a 46% reduction in power. However, we will see that this technique has a limited application depending on the chip timing response. The GLOBALFOUNDRIES libraries offer 3 different supply voltage options (0.9V, 0.8V and 0.65V).

3.3.1 Power domains definition

Voltage islands definition should be done very carefully in very frequency constrained designs as the one we are working on. For sure, not all the blocks need the maximum voltage supply to meet timing constraints, but find those who need it are the first task we should do.

In this design, the core unit has critical paths in the multiplier and divider. Moreover, some of the paths that pass trough this 2 modules have mutual connections with other core modules, so in general, there is not a good idea to decrease voltage in core modules. In fact, in general it is preferable not to split the main modules in different power domains.

The packetizer unit also has some of its modules close to the critical point, so it will be needed to keep it at the maximum voltage too.

Apart from this 2 modules, the rest of design can be lowered to 0.8V without considerable timing problems. However, other factors apart of timing should be taken into account. For example, the modules that share most of the connections with the core and packetizer should be packed together in the same domain. Otherwise, a very large amount of level shifters would be needed, compromising timing and area (even though area is not a key issue in this design). A level shifter cell can suppose a delay equivalent to put 3 or 4 inverters in series (a 22nm inverter has a typical delay of 10-15 ps while a level shifter can reach up to 45 ps). This means that paths that have some margin could even turn negative.

Another aspect to take into account is the module power consumption. Small modules often

have very low power consumption and lowering the voltage could induce that inserted level shifters use more power than saved.

With all of these factors, some modules have been encountered appropriated to decrease their voltage. All the submodules inside the ”Rocket” module (except the core unit) are susceptible to be lowered to 0.8V, while the peripheral modules, like the UART and the SPI are very low constrained and are susceptible to be lowered to 0.65V.

Due to some limitations in level shifter insertion that will be discussed in the next point, some Rocket submodules were unable to be lowered to 0.8V. With the prior assumptions, table 7 shows the final voltage islands and their respective modules.

Table 7: Voltage islands.

Voltage (V)	Modules
0.9V FB	default domain
0.8V FB	L2Cache/data JTAG L2-Net AddrConv SCM nastiPipe-Lite IO-Net mam_router
0.65V FB	UART AXI_PMU

The power intent has added power ports, power domains and power states. Nets VCC and VCL bring 0.8V and 0.65V, respectively. Body bias nets are divided now in 4, 2 of them for ±0.9V and the other 2 for ±0.8V. The new code is exposed below:

```
create_supply_port VCH
create_supply_port VCC
create_supply_port VCL
create_supply_port VSS
create_supply_port VCNW
create_supply_port VCPW
create_supply_port VCNW_L
create_supply_port VCPW_L

create_supply_net VCH
create_supply_net VCC
create_supply_net VCL
create_supply_net VSS
create_supply_net VCNW
create_supply_net VCPW
create_supply_net VCNW_L
create_supply_net VCPW_L

connect_supply_net VCH -ports VCH
connect_supply_net VCC -ports VCC
connect_supply_net VCL -ports VCL
connect_supply_net VSS -ports VSS
connect_supply_net VCNW -ports VCNW
connect_supply_net VCPW -ports VCPW
connect_supply_net VCNW_L -ports VCNW_L
connect_supply_net VCPW_L -ports VCPW_L

create_supply_set ss_aon_v09_bb  -function {power VCH}  -function {nwell VCNW} \
                                 -function {pwell VCPW}   -function {ground VSS}
create_supply_set ss_aon_v08_bb  -function {power VCC}  -function {nwell VCNW_L} \
                                 -function {pwell VCPW_L} -function {ground VSS}
create_supply_set ss_aon_v065_bb -function {power VCL}  -function {nwell VCNW_L} \
                                 -function {pwell VCPW_L} -function {ground VSS}
create_supply_set ss_VNW         -function {power VCNW} -function {ground VSS}
create_supply_set ss_VPW         -function {power VCPW} -function {ground VSS}
create_supply_set ss_VNW_L       -function {power VCNW_L} -function {ground VSS}
```

```
  create_supply_set ss_VPW_L       -function {power VCPW_L} -function {ground VSS}


  create_power_domain AO_V09_BB       -elements {.} -supply {primary ss_aon_v09_bb}
  create_power_domain AO_V08_BB       -elements {Rocket/L2HellaCacheBank/data \
                                                 Rocket/bsc_jtag Rocket/l2Network \
                                                 Rocket/nastiAddrConv \
                                                 Rocket/bsc_scm Rocket/nastiPipe \
                                                 Rocket/nasti_lite Rocket/ioNetwork \
                                                 Rocket/mam_router Rocket/scm_router} \
                                      -supply {primary ss_aon_v08_bb}
  create_power_domain AO_V065_BB      -elements {uart_i AXI_PMU_0}
                                      -supply {primary ss_aon_v065_bb}
  create_power_domain DUMMY_PD_VNW    -elements {} -supply {primary ss_VNW}
  create_power_domain DUMMY_PD_VPW    -elements {} -supply {primary ss_VPW}
  create_power_domain DUMMY_PD_VNW_L  -elements {} -supply {primary ss_VNW_L}
  create_power_domain DUMMY_PD_VPW_L  -elements {} -supply {primary ss_VPW_L}


add_power_state -supply AO_V09_BB.primary -state {h -supply_expr {power=={FULL_ON 0.9}\
                                                  &&  ground == {FULL_ON 0.0} \
                                                  &&  nwell == {FULL_ON 0.9} \
                                                  &&  pwell == {FULL_ON -0.9}} \
                                                  -simstate NORMAL }
add_power_state -supply AO_V08_BB.primary  -state {m -supply_expr {power=={FULL_ON 0.8
    }\
                                                  &&  ground == {FULL_ON 0.0} \
                                                  &&  nwell == {FULL_ON 0.8} \
                                                  &&  pwell == {FULL_ON -0.8}} \
                                                  -simstate NORMAL }
add_power_state -supply AO_V065_BB.primary -state {l -supply_expr {power=={FULL_ON 0.65
    }\
                                                  &&  ground == {FULL_ON 0.0} \
                                                  &&  nwell == {FULL_ON 0.8} \
                                                  &&  pwell == {FULL_ON -0.8}} \
                                                  -simstate NORMAL }

add_power_state AO_V09_BB             -state {PM1 -logic_expr {AO_V09_BB.primary == h \
                                              && AO_V08_BB.primary == m \
                                              && AO_V065_BB.primary == l}}
```

3.3.2 Level shifter insertion

Level shifters are low power cells that are essentially buffers that change from one voltage swing to another [6]. They are used in power domain boundaries to avoid logic voltage misunderstandings and over-heating.

When driving signals between power domains with radically different power rails, the need for level shifters is clear. Driving a signal from a 1V domain to a 5V domain is a problem. But the internal voltages in today's chips are tightly clustered around 1V. Why would we need level shifters on signals going from a 0.9V domain to a 1.2V domain?

One reason is that a 0.9V signal driving a 1.2V gate will turn on both the NMOS and PMOS networks, causing excessive crowbar currents. Crowbar currents are the ones produced in protection circuits that short-circuit the voltage supply when it exceeds the limits.

In addition, standard cell libraries are characterized for a clean, fast input that goes rail to rail. Failure to meet this requirement may result in signals exhibiting significant rise or fall time degradation between the driver and receiver in different voltage domains. This can lead to timing closure problems [1].

Level shifters are very often characterised for several scenarios. In 22nm libraries, high-to-low and bidirectional level shifters are available. Moreover, there are cells available to be placed in the source domain or in the destination domain. The decision of choosing the most adequate ones should be done according to the design characteristics. Source placed level shifters can satisfy multi-voltage nodes with fewer cells than destination placed ones, specially when module outputs have a large fanout.

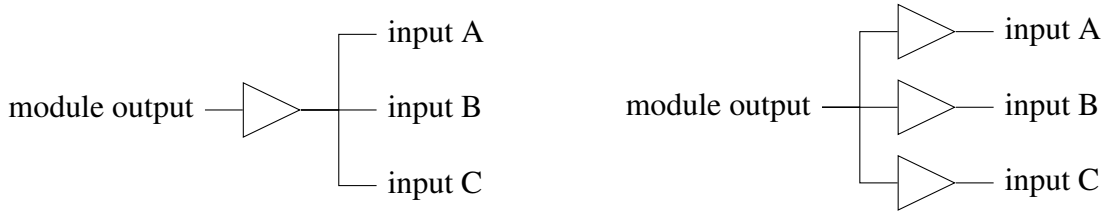


Figure 8: Source (left) and destination (right) level shifter comparison.

Despite of looking a bad decision, destination placed level shifters are very often the preferable option. The reason is that synthesis tools do not interpret rightly power intent definitions in source placed level shifters, making voltage errors at signal outputs. Figure 9 shows an error example.

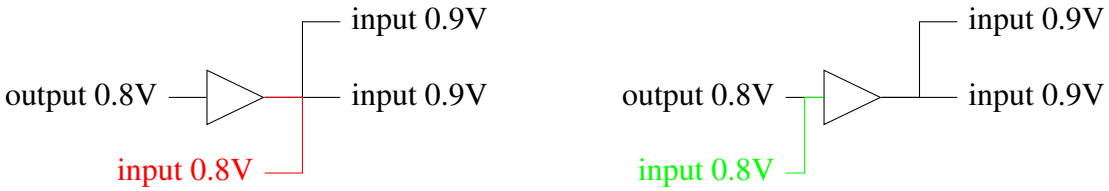


Figure 9: Source placed level shifter error. The tool connect level shifter output to input ports of the origin domain instead of using the input node. The left image shows the error and the right image, the correct interpretation.

A good strategy would be to try to place level shifters in the source domain and change to destination domain whose who report problems. However, in this work an additional problem has been to be faced. Unfortunately, not all the level shifting options were characterised for forward bias libraries, so **the design has been forced to be used only source placed level shifters**. This means that modules that were originally possible to be lowered their voltage had to be discarded when they reported level shifting problems. In addition, for low-to-high level shifters, the output driver has to make a greater effort than the input driver. In this case, put level shifters in destination domain is more appropriate.

Another issue was whether to use unidirectional level shifters or bidirectional ones in the case of high-to-low conversion. Unidirectional level shifters hare more robust but dissipate more power and suppose more delay, while bidirectional level shifters are for a general purpose (low-to-high or high-to-low). As no electrical constraints are reported for this work, bidirectional level shifters have been used to satisfy more easily timing constraints.

Level shifters are placed from different level shifting strategies specified in the UPF file [2]. The different options that are inside level shifting strategies are:

- ”-domain” specify the domain where level shifters belong.
- ”-source” specify the source domain.
- ”-location” specify possible locations. It can be ”self”, ”parent” or ”other”. In this work, we are going to use ”self” always.
- ”-elements” specify the modules inside the established domain where the strategy applies.
- ”-applies_to” specify if the strategy is applied to domain inputs, outputs or both.
- ”-rule” specify if the strategy is used for low-to-high, high-to-low or both level shifting options.

Some tips should be taken into account when designing level shifting strategies:

- Only one level shifting strategy can be done in a voltage island. For example, in this design, there 3 power domains (0.9V, 0.8V and 0.95V) and 2 voltage islands (0.8V and 0.65V). 0.9V is the default domain and does not need to be defined a voltage island. In this case, we can specify as much level shifters strategies in 0.9V as we want, but we can only define one level shifter strategy inside 0.8V domain and another one inside 0.65V domain. Otherwise the tool would report strategy errors. For that reason, we can not directly put level shifters from 0.8V to 0.65V. Paths that go from 0.8V to 0.65V has to pass by force to 0.9V domain. In other words, non default domains can only communicate directly with default domain.
- The UPF format choose automatically the most appropriate cell depending on the fanout. Then, the most useful strategy is to define all the compatible cells that could be used.
- When there are references to inputs or outputs in a strategy, they are referring to domain input-outputs, not module ones. Then, when we put level shifters in the destination domain we should use always ”applies_to inputs”, and when they are put in source domain, we should use ”applies_to outputs”.

The used strategies are exposed in the code below:

```
set_level_shifter lsh1 \  ##0.65V to 0.9V
   -domain AO_V065_BB \
   -source AO_V065_BB \
   -location self \
   -elements {uart_i AXI_PMU_0 PSO_MANAGER} \
   -applies_to outputs \
   -name_suffix _UPF_LS_LH_ \
   -rule low_to_high
use_interface_cell lvls1 -strategy lsh1 -domain AO_V065_BB -lib_cells {SC8T_LSSX1_CSC20SL
     SC8T_LSSX1_CSC24SL SC8T_LSSX1_CSC28SL SC8T_LSSX2_CSC20SL SC8T_LSSX2_CSC24SL
    SC8T_LSSX2_CSC28SL SC8T_LSSX4_CSC20SL SC8T_LSSX4_CSC24SL SC8T_LSSX4_CSC28SL}

set_level_shifter lsh2 \  ##0.9V to 0.65V
   -domain AO_V09_BB \
   -source AO_V09_BB \
   -location self \
   -elements {uart_i AXI_PMU_0 PSO_MANAGER} \
   -applies_to outputs \
   -name_suffix _UPF_LS_HL_ \
```

```
   -rule high_to_low
use_interface_cell lvls2 -strategy lsh2 -domain AO_V09_BB -lib_cells {SC8T_LSSX1_CSC20SL
    SC8T_LSSX1_CSC24SL SC8T_LSSX1_CSC28SL SC8T_LSSX2_CSC20SL SC8T_LSSX2_CSC24SL
    SC8T_LSSX2_CSC28SL SC8T_LSSX4_CSC20SL SC8T_LSSX4_CSC24SL SC8T_LSSX4_CSC28SL}

set_level_shifter lsh3 \  ##0.8V to 0.9V
   -domain AO_V08_BB \
   -source AO_V08_BB \
   -location self \
   -elements {Rocket/L2HellaCacheBank/data Rocket/bsc_jtag Rocket/l2Network Rocket/
       nastiAddrConv Rocket/bsc_scm Rocket/nastiPipe Rocket/nasti_lite Rocket/ioNetwork
       Rocket/mam_router Rocket/scm_router} \
   -applies_to outputs \
   -name_suffix _UPF_LS_LH_ \
   -rule low_to_high
use_interface_cell lvls3 -strategy lsh3 -domain AO_V08_BB -lib_cells {SC8T_LSSX1_CSC20SL
    SC8T_LSSX1_CSC24SL SC8T_LSSX1_CSC28SL SC8T_LSSX2_CSC20SL SC8T_LSSX2_CSC24SL
    SC8T_LSSX2_CSC28SL SC8T_LSSX4_CSC20SL SC8T_LSSX4_CSC24SL SC8T_LSSX4_CSC28SL}

set_level_shifter lsh4 \  ##0.9V to 0.8V
   -domain AO_V09_BB \
   -source AO_V09_BB \
   -location self \
   -elements {Rocket/L2HellaCacheBank/data Rocket/bsc_jtag Rocket/l2Network Rocket/
       nastiAddrConv Rocket/bsc_scm Rocket/nastiPipe Rocket/nasti_lite Rocket/ioNetwork
       Rocket/mam_router Rocket/scm_router} \
   -applies_to outputs \
   -name_suffix _UPF_LS_HL_ \
   -rule high_to_low
use_interface_cell lvls4 -strategy lsh4 -domain AO_V09_BB -lib_cells {SC8T_LSSX1_CSC20SL
    SC8T_LSSX1_CSC24SL SC8T_LSSX1_CSC28SL SC8T_LSSX2_CSC20SL SC8T_LSSX2_CSC24SL
    SC8T_LSSX2_CSC28SL SC8T_LSSX4_CSC20SL SC8T_LSSX4_CSC24SL SC8T_LSSX4_CSC28SL}
```

3.3.3 MSV design floorplan

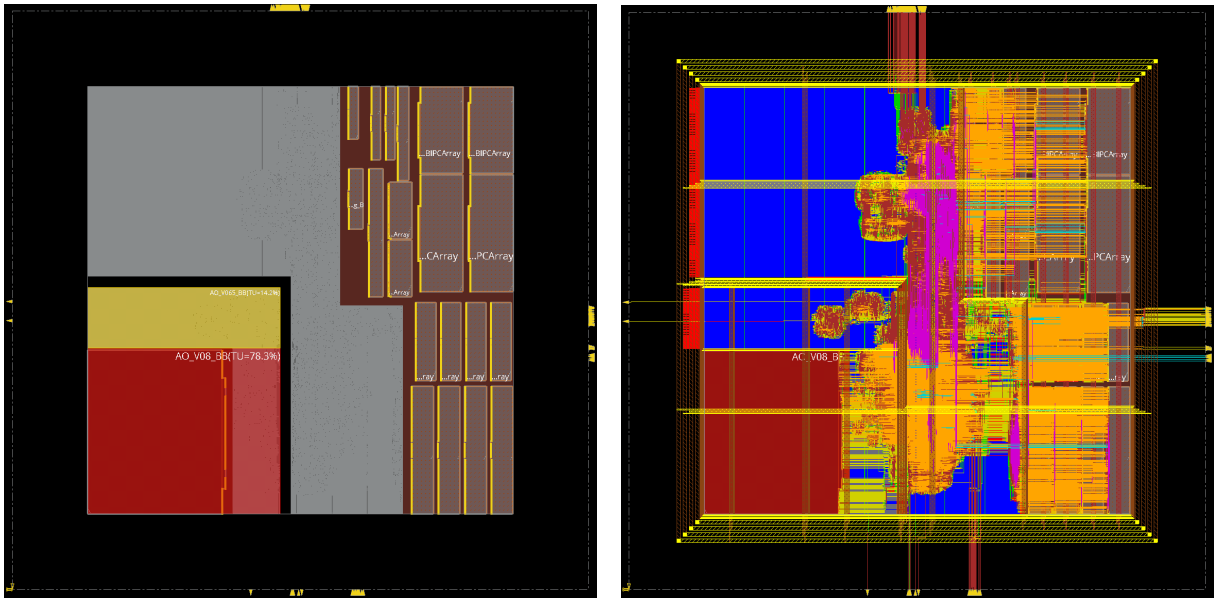


Figure 10: MSV Floorplan without routing on the left and with routing on the right.

The initial design floorplan has been needed to make some changes. As only one macro cell has been changed its operating voltage, no changes in macros disposition have been needed. Power

domains have been put in the inner left corner of the design to take advantage of the ”L2 cache” macro cell position. The 0.8V power covers ”L2 cache” area plus some std cell area in the left. The 0.65V domain covers the area over the memory. Both domains have been positioned in the same part of the chip to facilitate power planning and routing.

Respect to the power planning, applying multi-supply voltage techniques carries a substantial increase in area and complexity. The original 3-supply-port design has been changed to a 7-supply-port. Table 8 shows a summary of the new power rings.

Table 8: MSV power rings summary

Ring	Voltage (V)	Use	layer
VCH	0.9	Supply	JA JB (metal 9 and 10)
VCC	0.8	Supply	JA JB (metal 9 and 10)
VCL	0.65	Supply	JA JB (metal 9 and 10)
VSS	0	Ground	JA JB (metal 9 and 10)
VCNW	0.9	N bias	C1 (metal 3)
VCPW	-0.9	P bias	C1 (metal 3)
VCNW_L	0.8	N bias	C1 (metal 3)
VCPW_L	-0.8	P bias	C1 (metal 3)

The voltage gradient and distance between rails have been maximized in order to avoid disturbances. The ground rail has been intended to separate as much as possible power by placing it in the middle. A set of power rings of their respective domain have been inserted around macro cells and power domain boundaries. The reason behind this strategy is to have an isolated interface in each domain where vertical and horizontal power stripes can reach easily a limit of operation. Otherwise a greater number of power stripes would be needed in all the die. Figure 11 shows macro and power domain boundary rings distribution.

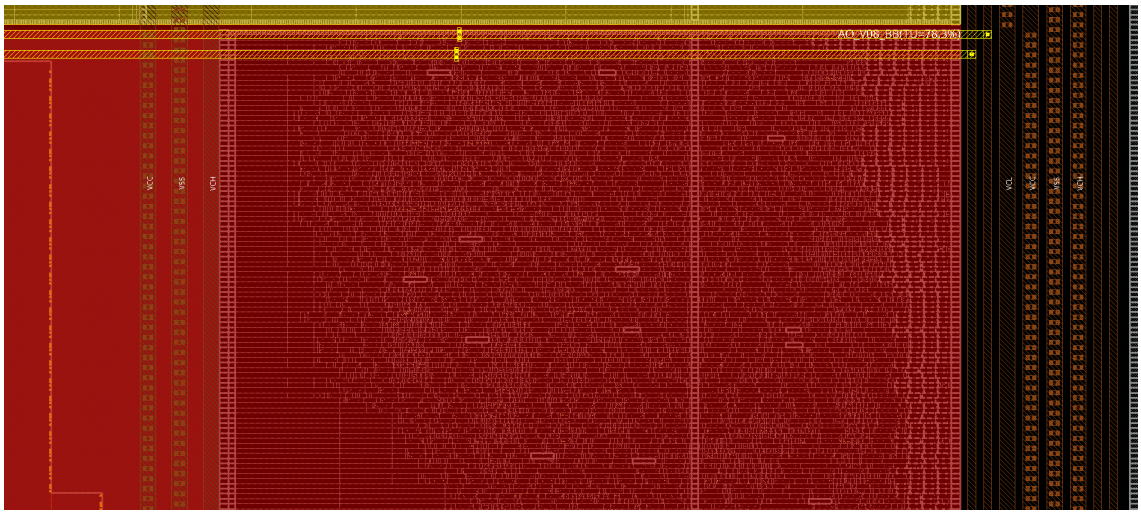


Figure 11: Space between L2 cache and 0.8V domain boundary.

Horizontal and vertical power stripes are located so as to have a minimal impact in chip congestion. For example, the horizontal stripes that crosses 0.8V and 0.9V domains are put at the same height and domain boundary ring changes supply stripes.

Respect to welltap cells, the distribution has been done similarly, but changing body bias rails depending on the power domain:

```
addWellTap -cell SC8T_TAPX8_CSC20L -cellInterval 70 -prefix WELLTAP_09
addWellTap -cell SC8T_TAPX8_CSC20L -cellInterval 70 -powerDomain A0_V065_BB -prefix
    WELLTAP_065
addWellTap -cell SC8T_TAPX8_CSC20L -cellInterval 70 -powerDomain A0_V08_BB -prefix
    WELLTAP_08
```

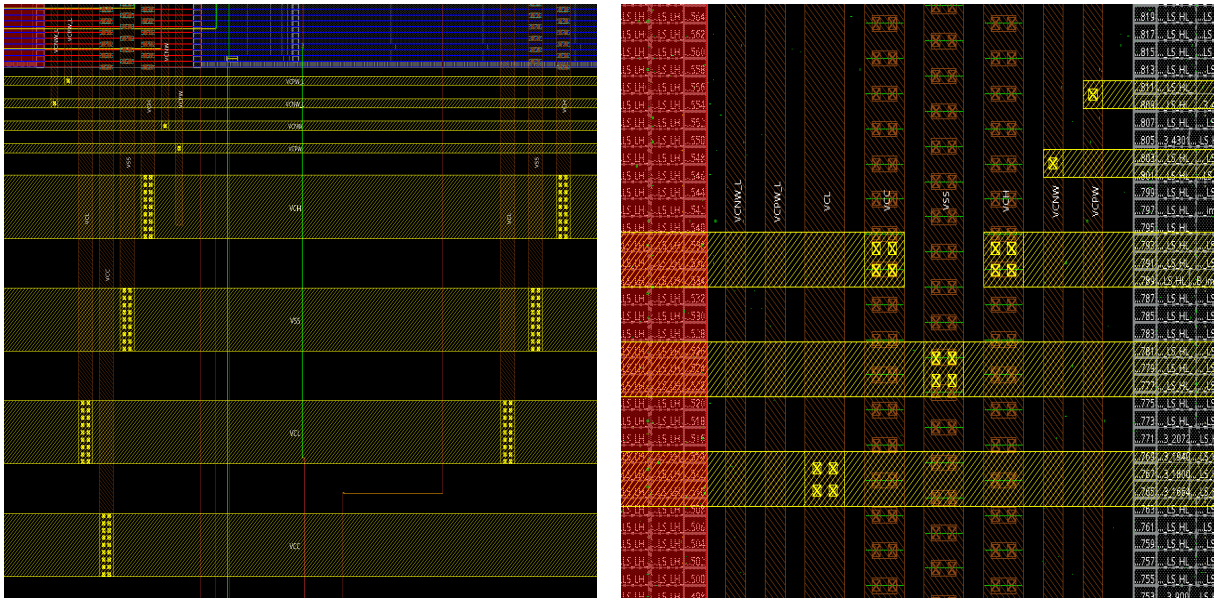


Figure 12: MSV power distribution.

3.4 Applying power shut-off design

Power gating consists in turn off chip modules when they are not used. The main purpose of this technique respect to clock gating is to reduce dynamic power but also leakage. If we compare with power optimization and multi-supply voltage, power gating is a more invasive technique in that affects inter-block interface communication and adds significant delays to safely enter and exit power gated modes [6].

Shutting down power to a module may be scheduled by control software as part of device drivers or idle tasks. Alternative it may be initiated in hardware by power management controllers. In our case, we focus in hardware implementation because we do not have access to software tools.

Some trade-offs are faced in power gating:

- The amount of possible leakage savings
- Entry and exit time penalties
- Power on energy used

In summary, power gating should not be done in a lot of cycles. The optimum point of this technique is when it is used in blocks that are not used for a long time (some microseconds in our 625 MHz case).

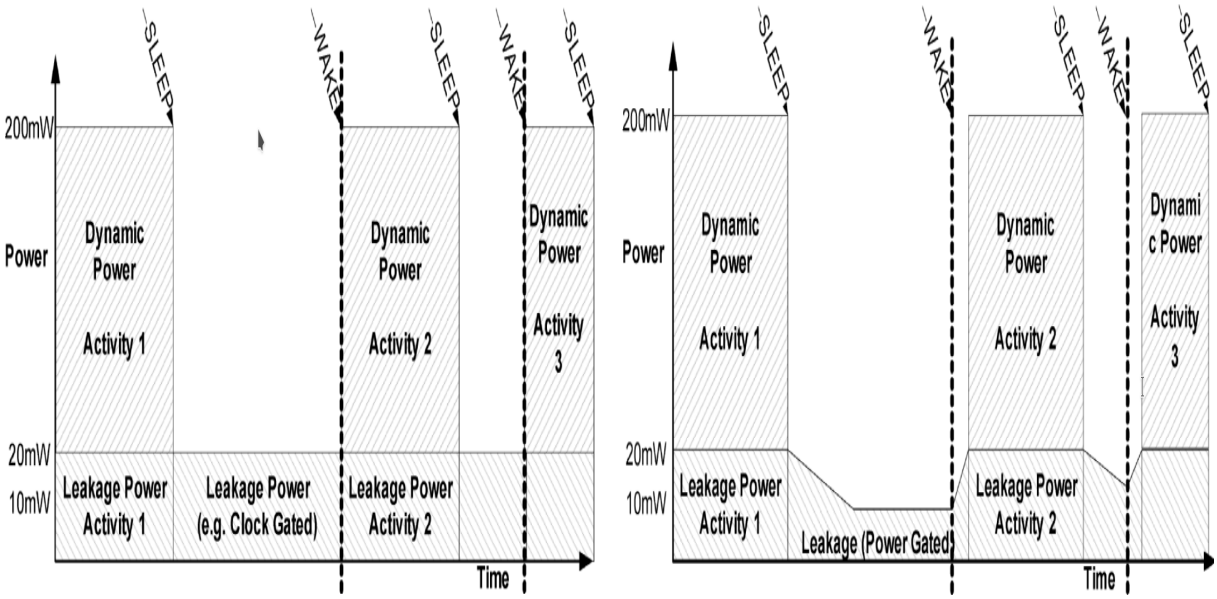


Figure 13: Clock gating (left) and power gating (right) power profile. Time delays are observed in leakage profile.

The best candidates for power gating in a processor design are usually duplicated modules and peripherals. This is for example, the second core unit of a dual core processor or the a USB unit when no information is sent or received. In our design there are not many opportunities. Only peripheral units are possible to shut off easily.

Peripheral units only represent a 1% of the total power. Moreover, only the write machine of the UART unit can be shut off for a long time. This means that power savings in this design could be null or even negative when applying power gating.

Nevertheless, the technique has been explored and applied in order to train for future designs.

3.4.1 Power gating strategy

The resulting power gated module consists in a FIFO chain used in the transmitter unit (TX_ASIC). Module inputs and outputs are described in table 9. No state machines or saved storage is needed in the module to keep the functionality. When the module is shut off, the only possible problem in terms of logic is that the stored data in the FIFO unit would be lost, but the power management strategy has been designed so as to be enough time in idle state so no information would be needed when shutting off. Therefore, no state retention registers are needed in this design.

The power shut-off strategy without retention registers is basically a simple power off and power of sequence that will be discussed deeply in the next chapter. Some requirements are needed for power gating without retention:

Table 9: TX FIFO module inputs and outputs. The default dormant value corresponds to the values that are displayed in idle state.

Type	Name	default dormant value
IN	data_i[7:0]	”00000000”
IN	space[7:0]	”00000000”
IN	push	0
IN	pull	0
IN	reset	0
OUT	data_o[7:0]	”00000000”
OUT	load	0
OUT	full	0
OUT	available[7:0]	”00000000”
OUT	available_write_space	1

To power gate a region without retention:

- Wait for any bus or external operations in progress
- 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 power down the block

To restore power:

- De-assert the power gating control signal to power back up the block
- De-assert reset to ensure clean initialization following the gated power-up
- De-assert the isolation control signal to restore all outputs

Figure 14 shows a summary of power control waveforms.

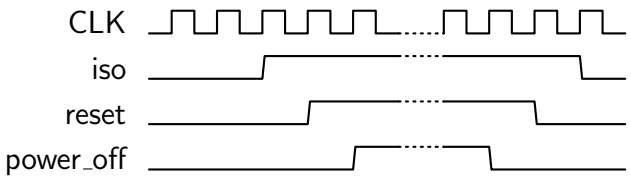


Figure 14: power-off and power-on sequence

Isolation cells are low power cells that are put in power gated modules outputs to isolate ”X” values when the module is shut-off. From a logic perspective, they are OR or AND gates that have outputs modules and a controlled isolation signal as inputs. AND gates are used to clamp outputs values to a logic ”0” when isolation signal is enabled. On the opposite, OR gates are used to clamp the output signal to ”1” [1].

To perform an isolation strategy, some module characteristics should be taken into account:

Figure 15: Basic isolation cells.

- The number of module outputs. A high outputs number would require isolation cells inside the power gated module to decrease the final number inserted cells, in the same way that has been explained about level shifters in the origin. However, putting isolation cells in a power gated module would require to keep them on while the module is off, which means that both always-on rail and powered shut off rail should be present in the switched domain, making the design more complex. Putting isolation cells in the origin also make easier to check the design because only one module outputs should be checked. As our design has just a few outputs, we have decided to put isolation cells in destination domain. As the power gated module is a FIFO UART submodule, it has only external connection with other UART submodules, which means that only one isolation strategy with UART domain has been needed.
- The clamp value of each output, which is the default value needed to keep the rest of the modules working without "X" propagation. In our case, all the output signals need a clamp value of "0" to keep the rest of modules in default idle states, so we have chosen an AND gate with an enable signal set to high (AONISOLOHX...).

In the UPF file, the isolation strategy has to be described as well as level shifter strategies [2]. location power domain, clamp value and elements are defined there. Some isolation cells with different fanouts are described to let the tool choose the most appropriate in each case.Moreover, isolation supply set marks the power rails that isolation cells should use:

```
set_isolation iso1_1 \
   -domain AO_V065_BB \
   -isolation_supply ss_aon_v065_bb \
   -isolation_signal "PSO_MANAGER/iso_en" \
   -clamp_value 0 \
   -elements uart_i/axi_uart_custom_instance/uart_core/uart_fifo_tx \
   -applies_to inputs \
   -name_prefix UPF_ISO_ \
   -name_suffix _UPF_ISO \
   -diff_supply_only FALSE \
   -location self
use_interface_cell isolation1 -strategy iso1_1 -domain AO_V065_BB -lib_cells {
    SC8T_AONISOLOHX1_CSC20L SC8T_AONISOLOHX1_CSC24L SC8T_AONISOLOHX1_CSC28L
    SC8T_AONISOLOHX2_CSC20L SC8T_AONISOLOHX2_CSC24L SC8T_ONOISOLOX2_CSC28L
    SC8T_AONISOLOHX4_CSC20L SC8T_AONISOLOHX4_CSC24L SC8T_AONISOLOHX4_CSC28L}
```

About power switches, so many options can be selected. The GF22FDX library package let us to choose between many different power switch architectures. Firstly, there are header and footer switches. Header switches are the ones that are disconnected to supply voltage while footer switches are disconnected to ground.

From an electrical point of view, header switches are often less efficient in switching than footer

switches, which means that they can result in a higher leakage and area.

From an architectural point of view, header cells are easily to understand and check. Moreover, in designs with level shifters like the one we are developing, switching the ground of a cell with 2 different power supplies could be a problem. Finally, signal integrity issues when clamping values to a logic 0 are avoided when using header switches [1].

As area is not a main concern in our design and we could have problems with level shifters and isolation cells in the case of using footers, header cells have been chosen.

Another decision to take is whether to do parallel or serial power switching or. There are many variants of power switching distribution:

- Fully parallel power switching. Some power switches are put in the domain and power up a specific group of cells each one. When the enabling signal changes, all the power switches change at the same time. This approach minimizes power up and power down time, but increase considerably in-rush current, which could cause high voltage spikes that could corrupt registers in always-on blocks.
- Fully serial power switching or daisy chain. Power switches have an enable input and a enable-out output which is activated once the power up or down sequence is finished. Every power switch enable input is connected with the output of the previous one, making a big chain. This solution makes switching time so much longer, but minimizes in-rush current.
- Multiple daisy chain power switching. It consist in a hybrid solution. Multiple daisy chains operate in parallel.

As no electrical information or checking tools were available at the moment of doing this work and no strict timing constraints were considered in the power gated module, the most conservative solution of daisy chain was adopted. Moreover, the biggest power switches were used in order to minimize IR-drop possible problems.

In the UPF file, power switching strategy specifies the location domain. Input supply port refers to the always on supply while output supply port refers to switched power rails. The control port refers to the signal used to enable power switching and the acknowledge signal is the output of the last power switch enabled, which means that the power up sequence as already finished. ON and OFF states are defined in function of power switch behaviour [2]. In this case, power switches are enabled with a logic 0, so the ON state is achieved with ”!EN”.

```
create_power_switch ps1 \
   -domain {SW_V065_BB} \
   -input_supply_port {VDDC VCL} \
   -output_supply_port {VDD VSW} \
   -control_port {EN PSO_MANAGER/power_off}  \
   -ack_port {ENOUT PSO_MANAGER/mode}  \
   -on_state {SW_V065_BB_on VDDC {!EN}} \
   -off_state {SW_V065_BB_off {EN}} \
   -switch_type coarse_grain
map_power_switch ps1 -lib_cells {SC8T4V_HEADERBUF600X80_CSC20L}
```

3.4.2 Power management controller

The power control has been approached with a RTL based module. The module architecture has been designed following some principles. First, it should be as simple as possible. The more complexity we put in the module, the more area and additional power it may use. Secondly, it should impact the minimum possible in the design. For that reason, the module has been placed in the 0.65V always-on domain, giving no additional timing or level shifting problems.

The module architecture is based in 3 structures:

- A Moore finite state machine, which is in charge of power management outputs for isolation cells and power switches.
- A set of registers connected in series that collect written data and detect when the initial sequence is performed.
- A 10-bit counter that gives information about the time in IDLE state.

The set of inputs needed to control the UART_TX channel are extracted from the main UART inputs and UART core internal variables. The following list explain in detail each module input:

- CLK: the ASIC main 625MHz clock is used in the power management controller.
- RSTN: the power management module uses the general reset input, which is the output of a common reset synchronizer.
- TX_idle: a special internal variable had to be created in the UART core. It specifies when the write state machine is in IDLE state (logic 1).
- ACK: it is the output signal of the last power switch buffer in the daisy chain. A logic 1 is received when all the power switched cells are turned on.
- w_data[31:0]: a general UART input that comes from the FPGA through the packetizer. The writing information and initial sequence come from this port.
- w_valid: specifies when the previous port is ready to be extracted the writing information. It is also a FPGA output coming through the packetizer.

The outputs are used to control the previously inserted low power cells (isolation cells and power switches), that work independently of the UART unit despite of being inserted inside UART submodules. Module outputs are described in the list below:

- iso_en: isolation cells are enabled or disabled by this signal. A logic 1 enables isolation while a logic 0 keeps the output value.
- reset_tx: an additional reset is added to the power gated module. It resets the module in the same way a general reset would do. It is added to the gated module with an OR function with the general reset.

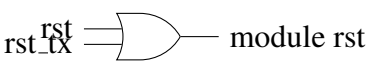


Figure 16: Power gated module reset.

- power_off: power switch cells are enabled or disabled with this variable. A logic 0 powers on the module while a logic 1 shuts it off.
- ACK_OUT: ACK inverted signal that is used in reset synchronization. The original ASIC reset has been needed to be replaced. The reason is that power switches take some time to power on (about 8 clock cycles), which means that the power gated module is remained off when the rest of modules are being initialized. This delay creates failures in UART synchronization. The proposed solution adds an additional step to the reset synchronizer. All the modules except the power management module have their reset delayed. The system waits until all the power switches are powered on.

```
always @(posedge clk or negedge rstn)
begin
  if(~rstn) begin
    rstn_2 <= 1'b0;
    rstn_1 <= 1'b0;
        rstn_PSO <= 1'b0;
  end else begin
    rstn_2 <= 1'b1;
    rstn_1 <= rstn_2;
    if(rstn_1 == 1'b1 & ~ack_out)
    begin
            rstn_PSO <= 1'b1;
    end
  end
end
```

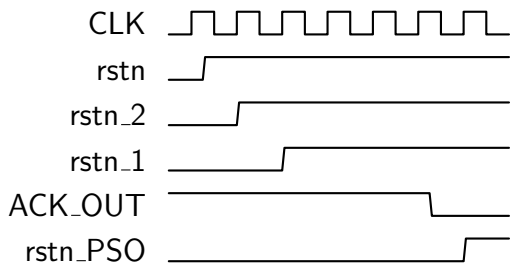


Figure 17: New reset synchronizer sequence. The ASIC reset signal (rstn_PSO) waits for a positive edge with ACK_OUT in 0.

The finite state machine works in 6 different states, each of them belongs to a power down or power up step.

The default on state is named ”iso-up”. All the isolation cells are disabled and power switches are On in this state. When the write_state UART machine enters in IDLE state, in means that no more information in sent to the ASIC, so a cycle counting starts. If the counter reaches 1000 consecutive clock cycles in IDLE state, the power down sequence starts.

Fist, the state ”iso-down” is reached. In this state, isolation cells are activated and, therefore, all the UART_TX outputs are clamped to 0. Just one clock cycle after isolation, if the IDLE variable keeps on, the FSM goes into the ”rst-on” state, where the UART_TX module is reset. To keep some time just in case the IDLE variable changes, the next state is entered when the counter marks another 5 clock cycles. After this, FSM goes to ”pwr-down” state, where all the switches are progressively switched off.

The power on sequence works inversely to the power off. First logic 1 is needed in the w_valid input to start the sequence. To start appropriately the UART unit, the initialization sequence should be done every time the UART has been shut off. After detect w_valid change, the FSM changes the state to ”pwr-up”, where the switches are progressively powered on. After all the switches are powered on, the input ”mode” should change from 0 to 1, resulting in another state change to ”rst-off” where reset signal in UART_TX is disabled. Just one clock cycle after, the

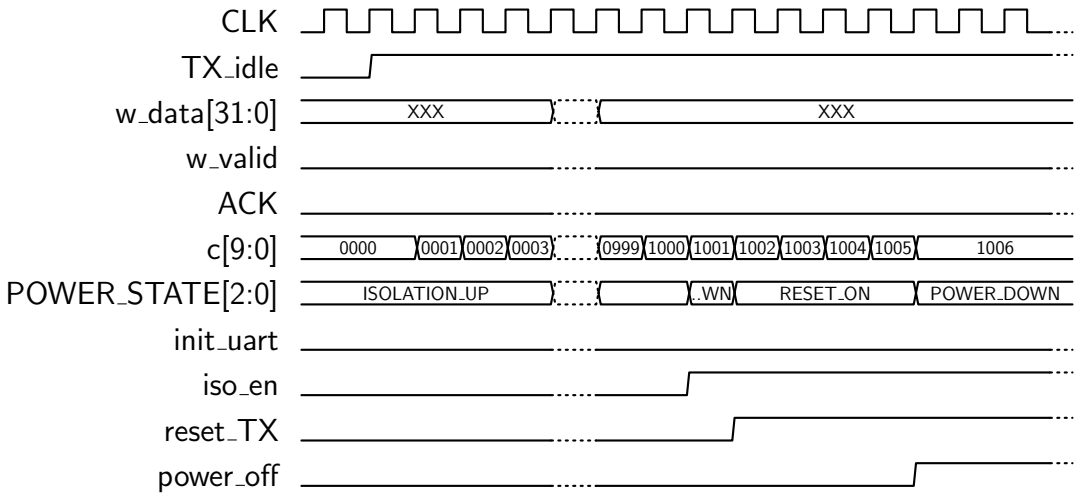


Figure 18: Power off power controller sequence.

FSM returns to the initial state and isolation signal is disabled, making the power gated module active again.

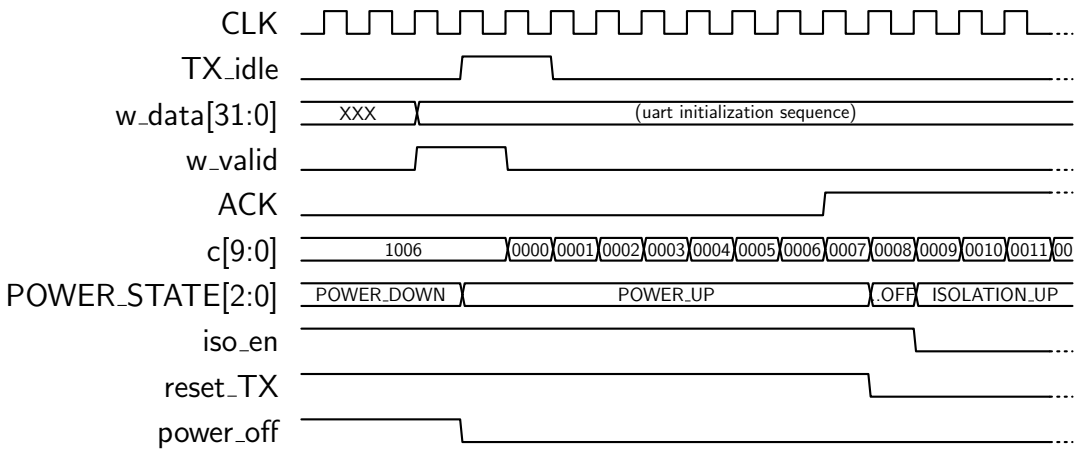


Figure 19: Power on power controller sequence.

The machine counter works when the IDLE variable is active and the FSM is in one of the 3 first states (iso-en, iso-down, rst-on). The counter is set to zero when the IDLE variable is inactive or when a general reset is active. Moreover, there is a array set in charge of detect when a the init sequence is performed. At least in the given examples, the initial sequence consist in a set of 6 hexadecimal numbers that are sent through the 9 LSB of w_data[31:0] (80, 1B2, 1B, 00, B and 1). This 6 arrays are always collected in the array set, which activates an auxiliary variable when this sequence is detected. While the auxiliary variable is active, the counter is stopped. This mechanism has been designed to prevent the UART unit to shut off after the initial sequence if a lot of commands are needed to be done before sending information.

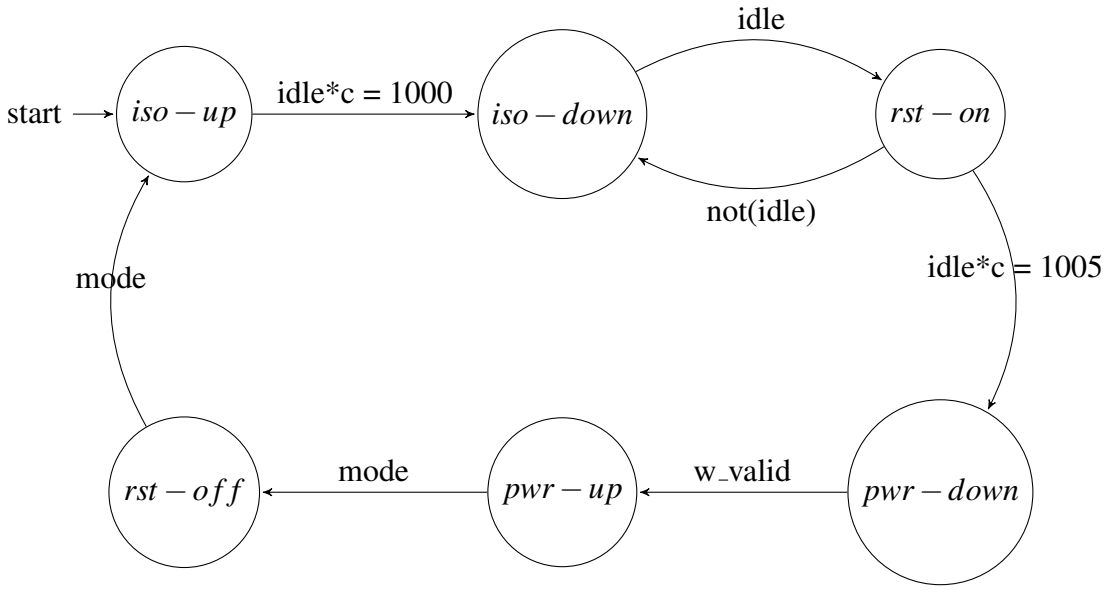


Figure 20: FSM state diagram

Table 10: FSM outputs in function of active state

output	ISO-UP	ISO-DOWN	RST-ON	PWR-DOWN	PWR-UP	RST-OFF
iso_en	0	1	1	1	1	1
reset_TX	0	0	1	1	1	0
power_off	1	1	1	0	1	1

3.4.3 PSO design floorplan

0.65V power domain has been split into 2 different domains for power switching. The UART_TX FIFO unit has been placed in the left in a switched 0.65V domain, while the rest of UART modules have been placed at the right to facilitate routing. Some changes in power grid network were also needed. A new switched power net was created called ”VSW”. Unlike other nets, the switched net has not been added to the core die rings. Instead, It covers the switched the switched 0.65V domain. The reason behind this approach is that VSW is actually a switched net that is feed by VCL in ON state and set to ground in OFF state. Thus, it is not necessary to put it in the core die.

About power switches distribution, Innovus tool let us choosing between 2 alternatives: ring distribution and column distribution. In ring implementation, a ring of switches surrounds the power domain and its directly connected to the always-on supply. In column distribution, some columns of power switches are placed inside the power domain.

The ring distribution has some advantages. First, it has a less complex power planning because switches do not disturb standard cell distribution inside the power domain. Moreover, it is the only way to shut-off hard blocks like memories.

However, some disadvantages are also present. It does not support retention register insertion

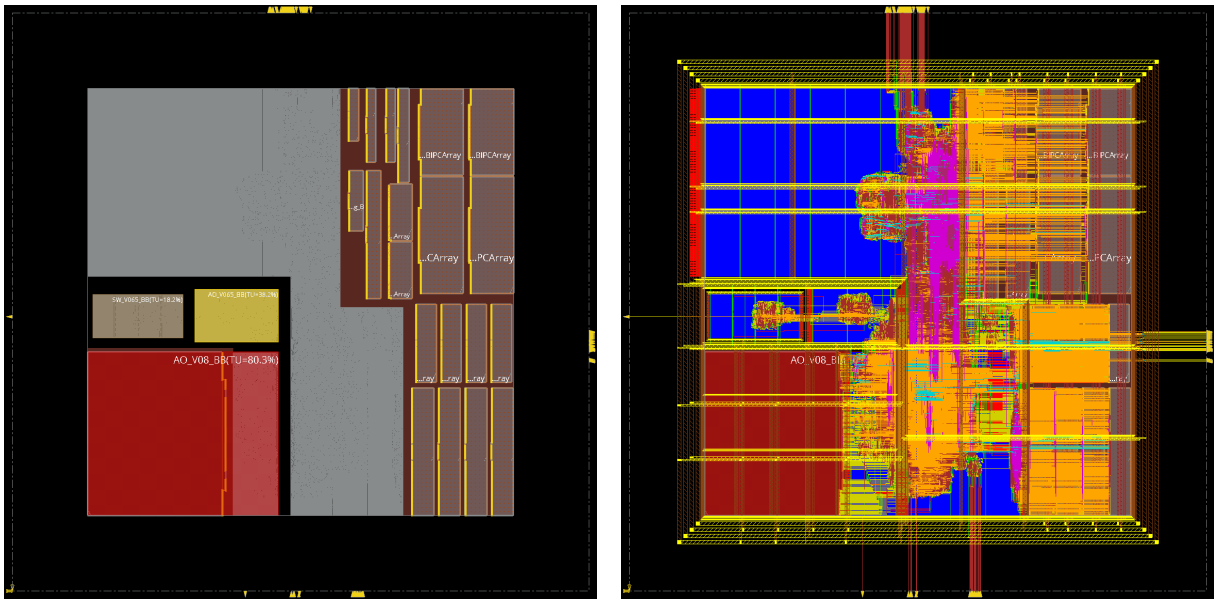


Figure 21: MSV-PSO Floorplan without routing on the left and with routing on the right.

and the area cost is often higher than in a column style implementation. Additionally, this approach offers less flexibility to distribute the power network, resulting in more difficulties to deal with IR-drop problems.

The column distribution offers more flexibility to distribute the power switching network by putting as much columns as we need. Moreover, the area impact is slower and state retention is possible. The main problem with this distribution is that adding additional columns in a design that already has welltap columns can affect considerably routing [1].

By looking the pros and cons of each distribution, a good conclusion is that ring style implementation is a better approach for small blocks like the one we are shutting down, while column style is better for bigger blocks without hard IPs. Then, the logical conclusion would be choosing ring style in our case. Nevertheless, in this work the column style distribution has been selected. The reason behind this conclusion is that this design is not a real approach. In fact, the real reason to do this power gating design is to train for future designs. As the future designs that the work group is facing are bigger than this and will need a more complex PSO, the column distribution will be necessary.

Power switches are inserted in P&R stage. The command used our case is the following:

```
addPowerSwitch -column -backToBackChain -powerDomain SW_V065_BB \\
-1801PowerSwitchRuleName ps1 -horizontalPitch 45 -leftOffset 45  \\
-globalSwitchCellName {SC8T4V_HEADERBUF600X80_CSC20L}  \\
-enablePinIn {EN} -enablePinOut {ENOUT} -enableNetIn {power_off} \\
-noFixedStdCellOverlap
```

The option ”column” defines the power distribution, ”backToBackChain” defines the full daisy chain of switch connection and ”enableNetIn” links the first power switch input to power manager controller control output. Figure 23 shows the switch enable distribution with arrows.

Finally, a static IR-drop analysis has been done to check switch congestion and power rail

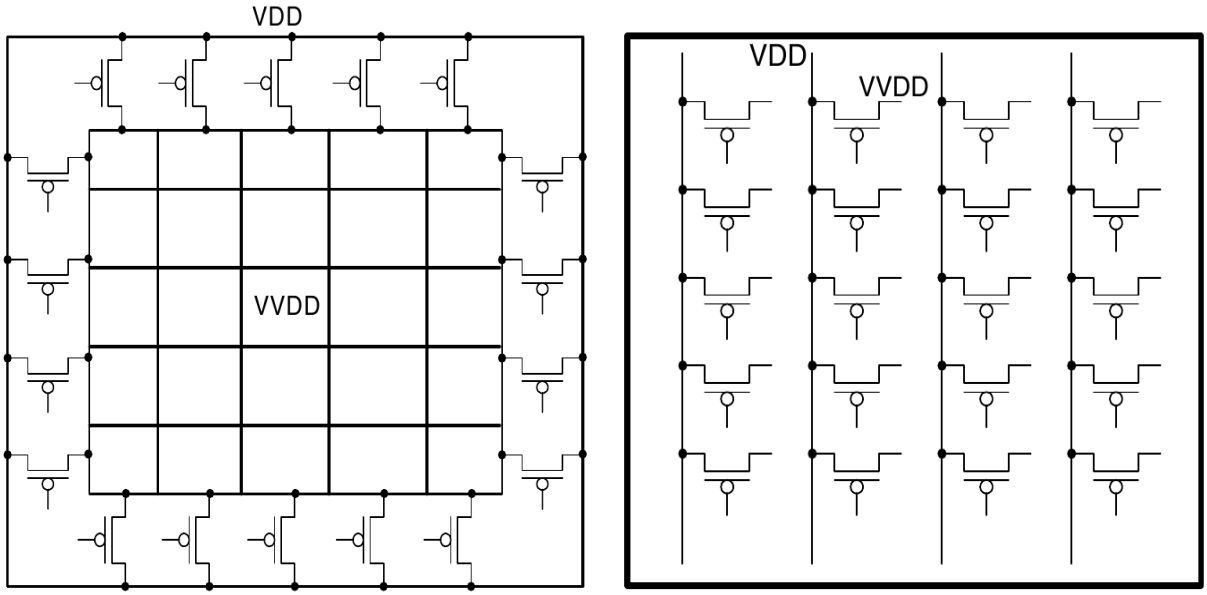


Figure 22: ring and column distribution

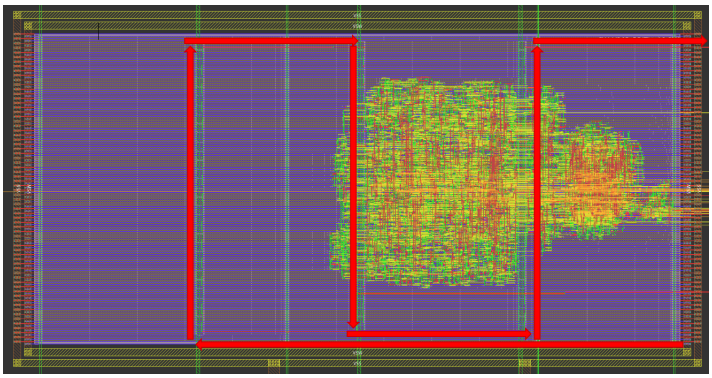


Figure 23: Power switch distribution.

distribution. Figure 24 shows the final check.

Almost all the modules in the design fulfil the IR drop $\pm 10\%$ rule in maximum range. Power switches appear in a clean green, which means that the design is even over-dimensioned. However, as no further dynamic check tools have been used, this power switch distribution has been kept to ensure a correct behavior. Only few parts of macro cells show parts in the close to range limit, but these parts have actually a low cell concentration, so in general terms the design is correct.

3.5 Internal power optimization

Once all the previous techniques have been approached and a reliable design is tested, we can activate internal synthesis tools to reduce power at gate level.

As we already know, power consumption in IC is divided in 2 types: dynamic power and leakage

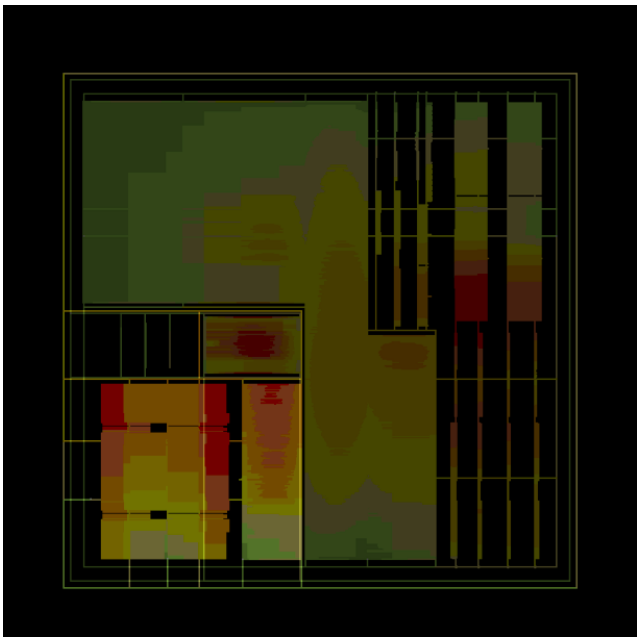


Figure 24: IR drop analysis.

power. Dynamic power is the power dissipated by an instantaneous short-circuit connection between voltage supply and ground when the gate transitions, and the gate power charging and discharging internal capacitances. Consequently, the dynamic power is calculated as follows:

$$P_d = C_L * V_{DD}^2 * f \tag{1}$$

Leakage power is the power dissipated by current leaks in the transistors. This value is often a constant of each library cell, which can be higher or lower depending on the threshold voltage. The higher the threshold, the lower the leakage.

Genus has low power commands that can reduce internally both leakage and dynamic power. Leakage power optimization is performed by changing low-threshold cells by high-threshold ones in non-critical paths. Dynamic power in reduced in 3 ways:

- Cell sizing: the tool can selectively increase and decrease cell drive strength throughout the critical path to achieve timing and then reduce dynamic power to a minimum
- Buffer and inverter removing and merging: the tool can identify unnecessary and repeated buffers and inverters, deleting and merging them.

The first step to perform an adequate power reduction is to provide switching information. By default, the Genus engine defines a 20% of switching activity in all the pins, which is not a realistic estimation [6]. To create a more realistic scenario, RTL toggle count stimulus files (TCF) files should be read after elaboration. This files have been created from the simulation models showed in previous chapters, making a data-sheet of 9 different stimulus. The commands used to read stimulus files are the following:

```
read_tcf ../TCF_RTL_625MHZ_PSO/1_RTL_625MHz_sim.tcf
read_tcf -update -weight 0.5 ../TCF_RTL_625MHZ_PSO/2_RTL_625MHz_sim.tcf
```

```
read_tcf -update -weight 0.5 ../TCF_RTL_625MHZ_PSO/3_RTL_625MHz_sim.tcf
read_tcf -update -weight 0.5 ../TCF_RTL_625MHZ_PSO/4_RTL_625MHz_sim.tcf
read_tcf -update -weight 0.5 ../TCF_RTL_625MHZ_PSO/5_RTL_625MHz_sim.tcf
read_tcf -update -weight 0.5 ../TCF_RTL_625MHZ_PSO/6_RTL_625MHz_sim.tcf
read_tcf -update -weight 0.5 ../TCF_RTL_625MHZ_PSO/7_RTL_625MHz_sim.tcf
read_tcf -update -weight 0.5 ../TCF_RTL_625MHZ_PSO/8_RTL_625MHz_sim.tcf
read_tcf -update -weight 0.5 ../TCF_RTL_625MHZ_PSO/9_RTL_625MHz_sim.tcf
```

The option ”weight” is used to determine the weighted importance of every model. In this case, a a general scenario, all the simulations have the same weight.

Dynamic and leakage power optimization is performed by using 4 basic commands:

```
set_attribute max_leakage_power <POWER IN nW> /<MODULE IN HIERARCHY>
set_attribute max_dynamic_power <POWER IN nW> /<MODULE IN HIERARCHY>
set_attribute lp_power_optimization_weight <BETWEEN 0 AND 1> /<MODULE IN HIERARCHY>
set_attribute lp_power_analysis_effort <low|medium|high>
```

The 2 first attributes define the maximum acceptable quantity of leakage and dynamic power of the design. In the case that the tool cannot fulfill the specs, it reports a warning message and power optimization is aborted, so this parameters are critical. The third parameter is needed for preference reasons. By default, Genus engine gives more importance to leakage power optimization. By choosing a percentage of dynamic vs leakage power weight in the final design, we can give more importance to dynamic power optimization. Finally, the last parameter stabilises the tool effort to maximize power reduction. A ”low” attribute means that area and timing should not be highly impacted, while a ”high” attribute maximizes power reduction without taking care of them.

A reasonable strategy to use this technique is by constraining the parameters to the prior default value as the maximum as an starting point. From there, some iterations could be tried to get the minimum achievable point. In this case, the weight has been limited ot a 13% of leakage and maximum dynamic and leakage powers of 174mW and 9mW, respectively. Moreover, as area is not a key factor in the design and timing is close to the limit point before power optimization, a ”high” value in the effort command has been chosen. Then, the final commands look as follows:

```
set_attribute max_leakage_power 9000000 /designs/Top_asic
set_attribute max_dynamic_power 174000000 /designs/Top_asic
set_attribute lp_power_optimization_weight 0.13 /designs/Top_asic
set_attribute lp_power_analysis_effort high
```

4 Final results

In this section, gate level simulation results will be exposed for every technique applied in the thesis work by order. First, multi-supply voltage results will be compared against initial design. Secondly, PSO will be discussed and finally, power optimization techniques.

4.1 Multi-supply voltage impact

Enabling different supply voltages in a chip voltage islands is commonly described as a high invasive technique. Several modifications in physical distribution have been needed. Some Rocket modules were decreased to 0.8V while PMU and UART were decreased to 0.65V. In terms of general power reduction, table 11 shows a summary of the results.

Table 11: Default gate level power.

Block	Rocket blocks power (mW)	UART-PMU power (mW)	Total power (mW)
Initial design	10.245	22.07	184.05
MSV design	4.903	20.37	178.1
Variation (%)	-52.14	-7.7	**-3.23**

As can be seen, a final power reduction of nearly a 3.3% has been achieved. This value is not as close to theoretical calculations. According to a first approximation, the block which decreases to 0.65V should be reduced about a 54% of its power while the block decreased to 0.8V should decrease about a 20%. This would mean a power reduction near to a 5.5%. However, 2 important factors have to been taken into account: level shifters and power stripes.

A total amount of 2517 bidirectional level shifters have been needed to make the design work. This level shifters suppose a power consumption of 2.58 mW and an additional area of $3562\mu m^2$. Only 54 of this level shifters are located on 0.65V power domain. For that reason, the power decrease is clearly closed to the ideal value in this block than in the 0.8V block.

Additionally, powering 2 additional voltages with different body bias means that 4 additional stripes have been needed for every columns and row in the design, resulting in almost twice of the original stripe area. This additional area has been derived in a greater congestion and, consequently, a more difficult routing, which justifies the rest of area and power increase.

In terms of area, the final design has an area of $0.3397mm^2$ while the original design has an area of $0.3347mm^2$. This means an **area increment of 1.5%.**

In terms of timing, the designs keep on fulfilling logical specs. The code below shows the worst 20 paths:

```
 Slack                 Endpoint                                                    Cost Group
-------------------------------------------------------------------------------------------
+0ps RocketTile/core/EXE_INTEGER_UNIT/INT_MUL_64B/stg1_result1_q_reg[95]/D clk_core
+0ps RocketTile/core/LATCH_EXE_WB/DATA_TO_CSR_reg[63]/D                    clk_core
+1ps RocketTile/core/EXE_INTEGER_UNIT/INT_MUL_64B/stg1_result2_q_reg[95]/D clk_core
+2ps RocketTile/dcache/s2_store_bypass_data_reg[63]/D                      clk_core
+2ps RocketTile/dcache/s3_req_data_reg[63]/D                               clk_core
+2ps RocketTile/core/EXE_INTEGER_UNIT/INT_MUL_64B/stg1_result1_q_reg[63]/D clk_core
```

```
+2ps RocketTile/core/LATCH_EXE_WB/DATA_TO_WB_reg[63]/D                      clk_core
+2ps RocketTile/core/EXE_INTEGER_UNIT/INT_MUL_64B/stg1_result1_q_reg[94]/D clk_core
+3ps RocketTile/core/EXE_INTEGER_UNIT/INT_DIV_64B/rh_q_reg[63]/D           clk_core
+3ps RocketTile/cache/icache/tag_array/IC_tag_array/AW[3]                  clk_core
+3ps RocketTile/dcache/data/T112/MDArray_tag_A/AW_A[2]                     clk_core
+4ps RocketTile/core/LATCH_EXE_WB/DATA_TO_CSR_reg[62]/D                    clk_core
+4ps nastiAddrConv/conv/cbase_update_1_reg[3]/SE                           clk_core
+4ps nastiAddrConv/conv/cbase_update_1_reg[1]/SE                           clk_core
+4ps nastiAddrConv/conv/cbase_update_1_reg[0]/SE                           clk_core
+4ps nastiAddrConv/conv/cbase_update_1_reg[4]/SE                           clk_core
+4ps nastiAddrConv/conv/cbase_update_1_reg[2]/SE                           clk_core
+4ps nastiAddrConv/conv/cbase_update_1_reg[7]/SE                           clk_core
+4ps nastiAddrConv/conv/cbase_update_1_reg[6]/SE                           clk_core
+5ps nastiAddrConv/conv/cbase_update_1_reg[5]/SE                           clk_core
```

4.2 Power-switch-off impact

As it has been mentioned earlier. The current design has few power-shut-off opportunities and the technique has been approached only to train it. Nevertheless, power impact will be discussed in this section. As a early result, power is no reduced with this technique. In fact, depending on the simulation it even increases. Table 12 shows a general summary of the results of the ”Hello World” test in gate level power.

Table 12: ”Hello World” gate level power.

Block	UART-FIFO-TX power (mW)	Total power (mW)
MSV design	1.603	132.96
PSO design	1.604	136.13
Variation (%)	+0.06	**+2.38**

The reasons for this power increase are mainly 2: power switches and power management unit. The power management unit adds 0.054mW. The final quantity of power switches with the chosen distribution is 95 cells, which means an additional power consumption of 1.52mW. Power switches also add an additional area of 505.65μm^2, which is a 22% of the power switched domain area. Moreover, some additional rings and stripes have been needed in the power gated block.

The final area of the power-switch-off design is 0.3403mm^2, which supposes an increment of a 0.18% of the MSV design.

In terms of timing, the designs keep on fulfilling logical specs. The code below shows the worst 20 paths:

```
 Slack                        Endpoint                                        Cost  Group
--------------------------------------------------------------------------------------
+0ps RocketTile/core/EXE_INTEGER_UNIT/INT_MUL_64B/stg1_result2_q_reg[95]/D clk_core
+0ps RocketTile/core/EXE_INTEGER_UNIT/INT_MUL_64B/stg1_result1_q_reg[95]/D clk_core
+0ps RocketTile/core/EXE_INTEGER_UNIT/INT_DIV_64B/rh_q_reg[63]/D           clk_core
+0ps ocketTile/core/EXE_INTEGER_UNIT/INT_DIV_64B/divisor_q_reg[63]/D      clk_core
+1ps RocketTile/core/EXE_INTEGER_UNIT/INT_MUL_64B/stg1_result1_q_reg[92]/D clk_core
+1ps RocketTile/core/EXE_INTEGER_UNIT/INT_MUL_64B/stg1_result1_q_reg[86]/D clk_core
+1ps RocketTile/core/LATCH_EXE_WB/DATA_TO_WB_reg[63]/D                     clk_core
+1ps RocketTile/core/EXE_INTEGER_UNIT/INT_MUL_64B/stg1_result1_q_reg[93]/D clk_core
+2ps RocketTile/core/LATCH_EXE_WB/DATA_TO_CSR_reg[63]/D                    clk_core
```

```
+2ps nastiAddrConv/conv/cbase_0_reg[10]/SE                                    clk_core
+2ps nastiAddrConv/conv/cbase_0_reg[6]/SE                                     clk_core
+2ps nastiAddrConv/conv/cbase_0_reg[7]/SE                                     clk_core
+2ps nastiAddrConv/conv/mask_0_reg[10]/SE                                     clk_core
+2ps RocketTile/core/EXE_INTEGER_UNIT/INT_MUL_64B/stg1_result1_q_reg[87]/D clk_core
+2ps nastiAddrConv/conv/cbase_1_reg[7]/SE                                     clk_core
+2ps nastiAddrConv/conv/mask_1_reg[10]/SE                                     clk_core
+2ps nastiAddrConv/conv/mask_1_reg[7]/SE                                      clk_core
+2ps nastiAddrConv/conv/cbase_1_reg[10]/SE                                    clk_core
+3ps nastiAddrConv/conv/mask_0_reg[7]/SE                                      clk_core
+3ps nastiAddrConv/conv/mask_2_reg[10]/SE                                     clk_core
```

4.3 Dynamic and leakage power optimization impact

Internal power optimization techniques are the less invasive and better reporting ones. Usually, by using this techniques the main drawback is the increase of area produced by cell resizing, but in our case the area has been even decreased to 0.3399 mm^2. The main reason for this unexpected good result is design architecture. The proposed processor is actually a first version of a future processor, which means that the RTL code is not definitive and then, is has quite margin of optimization. For that reason, inverter and buffer merging has done a greater work as usual, decreasing the area and optimizing power. Moreover, cell resizing has been very constrained because the design has very short timing margins. As a result, the main focus of the tool engine has been approached in leakage optimization.

In terms of timing, the designs keep on fulfilling logical specs. The code below shows the worst 20 paths:

```
 Slack                      Endpoint                                          Cost Group
--------------------------------------------------------------------------------------
+0ps RocketTile/core/EXE_INTEGER_UNIT/INT_MUL_64B/stg1_result2_q_reg[95]/D clk_core
+0ps RocketTile/core/EXE_INTEGER_UNIT/INT_DIV_64B/state_q_reg[0]/D         clk_core
+0ps AXI_PMU_0/AXI_PMU_inst/slv_reg_reg[5][31]/D                           clk_core
+1ps AXI_PMU_0/AXI_PMU_inst/slv_reg_reg[3][31]/D                           clk_core
+1ps RocketTile/core/LATCH_EXE_WB/DATA_TO_CSR_reg[63]/D                    clk_core
+1ps RocketTile/core/EXE_INTEGER_UNIT/INT_DIV_64B/rh_q_reg[63]/D           clk_core
+1ps tdo                                                                   clk_core
+1ps RocketTile/core/LATCH_EXE_WB/DATA_TO_WB_reg[63]/D                     clk_core
+1ps RocketTile/dcache/s3_req_data_reg[63]/D                               clk_core
+1ps RocketTile/core/EXE_INTEGER_UNIT/INT_MUL_64B/stg1_result1_q_reg[95]/D clk_core
+2ps L2HellaCacheBank/bsc_mam/fifo_out/buffer/dout_reg[7]/D                clk_core
+2ps RocketTile/dcache/s3_req_data_reg[62]/D                               clk_core
+2ps uart_i/axi_uart_custom_instance/uart_core/uart_controller_i0/uart_transmitter_i0/
    bitcount_reg[3]/D                                                     clk_core
+2ps RocketTile/core/LATCH_EXE_WB/DATA_TO_CSR_reg[59]/D                    clk_core
+2ps RocketTile/dcache/s2_store_bypass_data_reg[63]/D                      clk_core
+3ps RocketTile/cache/icache/tag_array/IC_tag_array/AW[1]                  clk_core
+3ps RocketTile/core/LATCH_EXE_WB/DATA_TO_WB_reg[57]/D                     clk_core
+4ps RocketTile/dcache/s2_store_bypass_data_reg[61]/D                      clk_core
+4ps RocketTile/core/LATCH_EXE_WB/DATA_TO_CSR_reg[57]/D                    clk_core
+4ps RocketTile/core/LATCH_EXE_WB/DATA_TO_WB_reg[59]/D                     clk_core
```

Finally, Table 13 shows a summary of the power results.

As a final summary, table 14 shows power progression with all the different gate level simulations.

Table 13: Default gate level power.

	Leakage power (mW)	Dynamic power (mW)	Total power (mW)
PSO design	26.01	151.92	177.93
DLPO design	7.21	143.45	150.66
Variation (%)	-72.3	-5.57	**-15.32**

Table 14: Default gate level power.

Test	0.9V (mW)	MSV (mW)	PSO (mW)	DLPO (mW)	Variation (%)
Default	184.05	178.1	177.93	150.66	-18.14
CSR	137.6	134.91	135.29	114.36	-16.88
ILLEGAL	137.7	134.97	135.32	114.47	-16.87
ADDR	138.21	135.46	135.81	114.89	-16.87
FETCH	137.93	135.02	135.39	114.52	-16.97
SBREAK	137.65	134.95	135.3	114.46	-16.84
SCALL	137.67	134.95	135.31	114.46	-16.85
TIMER	139.5	136.82	137.18	116.23	-16.68
HELLO	135.61	132.96	136.13	114.94	-15.24

5 Conclusions

Respect to the initial timing and area goals of the migration, 22nm technology gives an additional range compared with 65nm technology. The design fitted easily in the requested area with a wide margin, meaning that additional low power techniques have been more easy to be implemented with this technology. Moreover, low power cell libraries are more complete, giving better opportunities and optimizations. Respect to timing, the migration has increase the operating frequency from the original 200MHz at 65nm to 625MHz. The RTL is still needed to be optimized, but only with a technology jump frequency has been increased by a factor of three, which is a big step.

In the present work, 3 different tools to reduce power consumption in ASICs have been tested. Changing the supply voltage in some regions has been probed as possible way to reduce power. However, some difficulties in level shifter insertion and a very constrained design in timing limited the efficacy of the technique. Moreover, some memory hard blocks were not able to be characterized for lower voltages. The technique has been known to be very efficient to low constrained modules with few input and output ports, for example, peripherals. With this in mind, is possible to say that static multi-supply voltage can have power reductions margins between 3% and 10% assuming area increases between 2% and 5% in single-core RISC-V processors like the one we have in this work.

power shut-off have been probed to be more difficult to implement in single-core processors. A key aspect to make this technique worth is to look to modules with a very restricted use than can be supposed to be a long time (greater than milliseconds) without been used (secondary core units, peripheral ports in wireless devices). For example, the future version of the Lagarto chip will have a vector processing unit composed of 2 independent modules. Moreover, the power switch distribution used in this works shows the most conservative way to perform power gating. As it has been exposed, a large area impact and routing problems could appear in power gated modules by defining a daisy chain. For that reason. Power gated modules should overcome easily power switch power consumption. In the presented work, the UART FIFO unit was composed only by approximately 2000 standard cells. A big and robust power switch like the ones we have used in this work can easily have a dynamic power consumption equivalent to 15 inverters. Future designs should use fewer and smaller power witches to ensure power reduction.

Respect to power optimization, the brief exposition in this work shows that it is the easiest and more efficient way to reduce power. In future designs that are ready to be tested, this technique could be very useful as a quick optimization way to meet system specs. Power reductions greater than 10% can be easily achieved by only applying 4 commands. However, in very area constrained design it could be problematic.

References

[1] David Flynn, Rob Aitken, Alan Gibbons, and Kaijian Shi. *Low power methodology manual: for system-on-chip design*. Springer Science & Business Media, 2007.

[2] IEEE Computer Society, 3 Park Avenue New York, NY 10016-5997USA. *IEEE 1801 Standard for Design and Verification of Low-Power, Energy- Aware Electronic Systems*, September 2018.

[3] J. Abella, C. Bulla, G. Cabo, F. J. Cazorla, A. Cristal, M. Doblas, R. Figueras, A. González, C. Hernández, C. Hernández, V. Jiménez, L. Kosmidis, V. Kostalabros, R. Langarita, N. Leyva, G. López-Paradís, J. Marimon, R. Martínez, J. Mendoza, F. Moll, M. Moretó, J. Pavón, C. Ramírez, M. A. Ramírez, C. Rojas, A. Rubio, A. Ruiz, N. Sonmez, V. Soria, L. Terés, O. Unsal, M. Valero, I. Vargas, L. Villa, and C. Ramírez. "an academic risc-v silicon implementation based on open-source components". In *2020 XXXV Conference on Design of Circuits and Integrated Systems (DCIS)*, pages 1–6, 2020.

[4] Cadence Design Systems, Inc. *Full Flow RAK - RTL to Place Route, Including ECO. Rapid Adoption Kit (RAK)*, July 2017.

[5] GLOBALFOUNDRIES. *22FDx Digital Design Guidelines*, February 2019.

[6] Cadence Design Systems, Inc. *Genus Low Power for Legacy UI*, August 2019.

[7] Joerg Winkler and Tamer Ragheb. "Implementation of ARM®Cortex ®-A17 Quad-core in GLOBALFOUNDRIES 22FDXTM technology using Cadence Innovus". GLOBALFOUNDRIES, 2017.

[8] Cadence Design Systems. Setting up body-bias voltage-based timing interpolation for fdsoi in innovus and tempus. support.cadence.com.

[9] Cadence Design Systems, Inc. *Power Analysis Flow from RTL Power through Signoff Power with Joules and Voltus*, February 2019.

[10] Cadence Design Systems, Inc. *SimVision Introduction*, September 2019.

[11] Cadence Design Systems, Inc. *Innovus User Guide*, December 2019.

[12] Cadence Design Systems, Inc. *Genus User Guide*, February 2020.

[13] GLOBALFOUNDRIES. *Standard Cells 8T User Guide*, January 2019.