**Omar Sakr**

**Master's of Nanotechnologies for integrated circuits**
**2017/2018**

**NXP Semiconductors**
**45 allée des ormes, Mougins, 06250**

# i.MX8 clock and power management

from 19/02/2018 to 17/08/2018

**Under the supervision of:**

- **Company supervisor: Olivier Milou** **email: olivier.milou@nxp.com**
- **Phelma Tutor: Lorena Anghel** **email: lorena.anghel@grenoble-inp.fr**

Confidentiality: yes

## Abstract

The i.MX8 is a series of NXP applications processors mainly dedicated for automotive applications. The architecture of these processors involves having in each subsystem a clock generator block, called Distributed Slave Controller (DSC). Currently, the DSC is hardened in the design and contains many hardened modules at different hierarchical level, which is suspected to induce high clock latency of the block and leads to performance limitation and leakage and dynamic power increase. So, the aim is to redesign the DSC with a different strategy to decrease the latency. As a first step, the slices, which are the clock dividers in the DSC, were redesigned completely flat without any hardening. Then they were implemented directly in the CCI subsystem (taken as an example) with another module as hard Macros and the rest of the DSC was flattened. As a result of this new design strategy, the latency decreased by the 30%. Therefore, the leakage power decrease by 50% for the slices and 10% for the subsystem and the area decreased by 15% for the slices and 7.5% for the CCI.

L'i.MX8 est une famille de processeurs de NXP principalement dédiés au domaine de l'automotive. L'architecture de ses processeurs met en place un block générateur des signaux d'horloge, appelé Distributed Slave Controller (DSC). Actuellement, le design du DSC est implémenté comme Hard Macro et il en contient beaucoup à différents niveaux hiérarchiques, ce qui induit une latence d'horloge élevée et entraine par conséquent une limitation de performance et augmente la fuite de puissance ainsi que la consommation dynamique. Alors le but du stage est de refaire le design du DSC avec une stratégie différente pour diminuer la latence. En première étape, les blocks diviseurs d'horloge sont refaits complètement à plat sans aucune Hard Macro. Ensuite, ils

sont implémentés directement dans le sous-système CCI (Pris comme exemple) avec un autre module en tant que Hard Macro et le reste du DSC est mis à plat. Grace à cette nouvelle stratégie, la latence a diminué de 30%. Par conséquent, la fuite de puissance a diminué de 50% pour les diviseurs et de 10% pour le sous-système. L'aire a diminué de 15% pour les diviseurs et de 7.5% pour le CCI.

L'i.MX8 è una famiglia di processori prodotti da NXP, principalmente utilizzati nelle applicazioni in ambito automotive. L'architettura di questi processori fa si che in ogni sottosistema vi sia un blocco per generare il clock, chiamato Distributed Slave Controller (DSC). Attualmente il DSC è implementato tramite una macro a diversi livelli gerarchici, ciò introduce un'elevata latenza nel segnale di clock del blocco con un conseguente peggioramento delle prestazioni in termini di performance e potenza. L'obiettivo è di ridisegnare il DSC con una strategia diversa per ridurre la latenza. Come primo passo, gli "slices" che dividono il clock nel DSC, sono state riprogettati completamente senza l'utilizzo di macro. Successivamente sono stati implementati direttamente nel sottosistema CCI (utilizzato come campione) inglobando la logica del DSC nel sottositema con la nuova versione degli "slices". Come risultato di questa nuova strategia di progettazione, la latenza è diminuita del 30%, la dissipazione in termini di potenza si è ridotta del 50% per gli "slices" e del 10% per il sottosistema con un decremento dell'area pari al 15% per i primi e del 7,5% per il modulo CCI.

## Acknowledgements

Before getting into the details of the work I did for 6 months, I would like to thank the people whom help made this experience happens in the best way that allowed me to evolve and learn.

Many thanks go to my NXP tutor, Olivier MILOU, senior design engineer, for all the things he taught me during my internship, for all the advises he gave me and his well guidance of the project. I would like to thank him also for his patience in answering to all my questions and for all his explanations. Besides I appreciate his help in my integration to the team.

I thank also Fabien JUMEL, the SOC design team manager, for recruiting me for this internship and for giving me the chance to have this great experience that will be a main building block of my professional career. I am also grateful for his advises on different phases and deliverables of the project.

I would like to thank a lot all the members of the backend design team, first for integrating me and making me feel a real part of the team, second for their help on the technical issues, their explanations and to allow me to learn from their expertise and their experiences.

I am thankful also for my all colleagues at NXP for the welcoming and joyful atmosphere in which I worked during the past 6 months. It has been a remarkable period for me.

I thank also Professor Lorena ANGHEL for her time to review and follow-up with me on my work and her remarks on the report.

## Table of Contents

## List of figures

## List of tables

| Abbreviations | |
|---|---|
| RTL | Register Transfer Language |
| STA | Static Timing Analysis |
| DRV | Design Rule Violation |
| DRC | Design Rule Check |
| AON | Always ON |
| PG | Power Gated |
| LVS | Layout Versus Schematic |
| ECO | Engineering Change Order |
| DSC | Distributed Slave Controller |
| SCU | System Control Unit |
| MSI | Medium Speed Interface |
| PnR | Place And Route |
| CCI | Cache Coherence Interface |
| CTS | Clock Tree Synthesis |

# 1. Introduction

## 1.1. NXP semiconductors

NXP is a Dutch company which is a leader in the semiconductors field. It designs and manufactures a wide variety of products in the application areas of connected cars, security, wearable electronics and Internet of Things (IOT). The company has 31,000 employees in more than 33 countries and a posted revenue of $9.5 billion in 2016. [1]

The internship was held in the French R&D site of Mougins. The main activities of the site are the design of microcontrollers with embedded microprocessors (mainly for automotive applications), security solutions based on NFC chips which targets online and wireless payment applications, and connectivity solutions with advanced protocols integrated on chips for IOT applications. The site has around 250 employees divided on 3 business units, each one of them works on a specific high-end application. The internship took place in the SOC back-end team of the MICR business unit which works on microcontrollers design. The team's role is to receive from the front-end team the RTL codes modeling the circuits to be designed at the behavioral level and to perform tasks going from synthesis to physical design, which results in transforming the codes into a physical layout that can be sent to the foundry for manufacturing. This process consists of many steps with various tools for either design or performance analysis (for power consumption, timing analysis, …). The latter is important to make sure the design corresponds to the chip specifications.

## 1.2. i.MX8 microcontrollers

The i.MX8 series of applications processors is a feature- and performance-scalable multi-core platform that includes single-, dual-, and quad-core families based on the Arm® Cortex® architecture including combined Cortex-A72 + Cortex-A53, Cortex-A35, and Cortex-M4 based solutions for advanced graphics, imaging, machine vision, audio, voice, video, and safety-critical applications.

i.MX8QM is a microcontroller that belongs to this family and is the subject of this internship project. It targets high end automotive and industrial market segments. It is built in 28 FDSOI leading edge technology to achieve both high performances and low power consumption [2].

Each device of the family has one switch matrix where all subsystems are plugged-in, a SCU (System Controller Unit) which manages the subsystems of the device, up to 16x subsystems (including SCU) and one/two DRAM controller(s). This architecture allows the devices to be scalable, as any subsystem can be replaced easily. This involves only disconnecting it from the switch matrix and connecting the new subsystem. This also gives an easiness in designing all the devices of the family. The SCU has an upper-hand control over the subsystems concerning boot, power, clock and reset management as well as resource partitioning / access control. All those mechanisms are abstracted thanks to a system control firmware running in the SCU. [3]

Each subsystem contains a block called the DSC (Distributed system controller). It provides power management and clock control to the subsystem. It is then the local controller used by the SCU to manage the different subsystems. The SCU interfaces with all the DSCs in the device through an MSI bus. The DSC embeds all the control logic to power up and power down domain(s) of a subsystem. It has reset register to control reset of power domain(s). It includes PLL(s) and clock dividers that provides source clocks to IP blocks of a subsystem according to the needs of the latter [3].

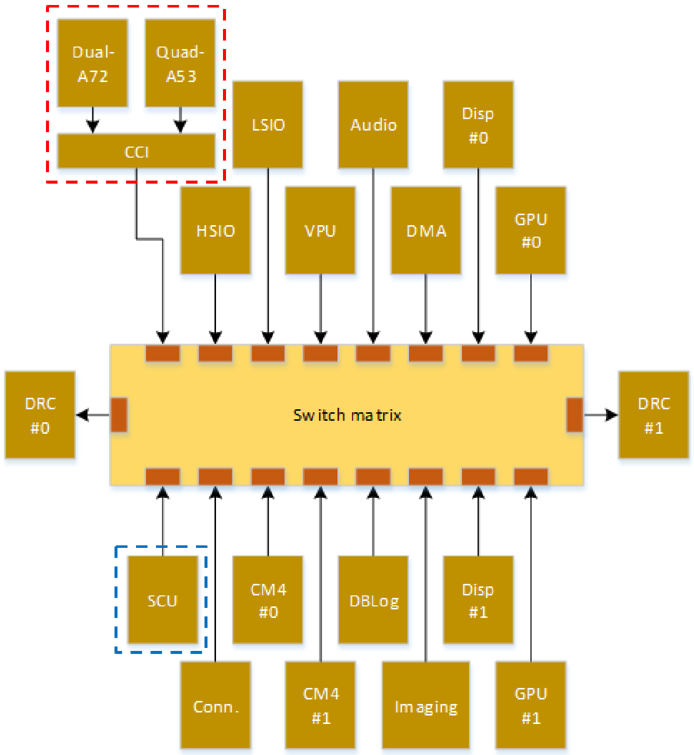


Figure 1: i.MX8 family architecture [3]

## 1.3. Internship objective

The main objective of the internship is the analysis and the optimization of the DSC block. The DSC is an important IP for the company for different reasons. First, it has been used in all the i.MX8 chips. Second, it is used in all the subsystems, which means it has a global effect on the whole chip. And last, it has a critical role from a functional point of view. It is responsible for clock generation to all the subsystems, which means it directly affect the performance/speed and the total power consumption of the chip. Therefore, the optimization of the DSC is an important issue for the future projects of the business unit.

The main issue of the actual design of the DSC is the high clock latency between the input and the output, which means the time between the arrival of clock rising edge to the DSC and the generation of an equivalent rising edge is too high. Decreasing the DSC's latency will lead to enhancing the performance of the subsystems and will allow them to run faster. In addition, optimizing the clock latency results in a decrease of the number of hold violations created on the subsystem level, which leads to a decrease of hold buffers needed at the subsystem level to fix hold violations and thus decreases the subsystem consumption. As the DSC is contained in all the subsystems, this decrease leads to an overall decrease of the power consumption of the chip, which is an important benefit. But also, the decrease of the number of violations leads to less work in the subsystems design and a shorter development time, which is not a negligible benefit. These elements further explain why this internship subject is important for the company.

The suspected technical reason behind the high latency of the DSC is the use of too much hardening in its design. Hardening means that some modules are processed by the full backend design flow (synthesis and place & route), then they are implemented as a one-unit cell in the higher level of hierarchy (like the technology library cells). This is then repeated at different level of the hierarchy of the DSC. This strategy is suspected to prevent the backend design tools from optimizing the timing, which results in the high latency. The objective of the internship is then the analysis of these modules to understand the relevance of this hardening and to adopt different design strategies to decrease the latency of the DSC.

## 1.4. Project Schedule and Gantt diagram

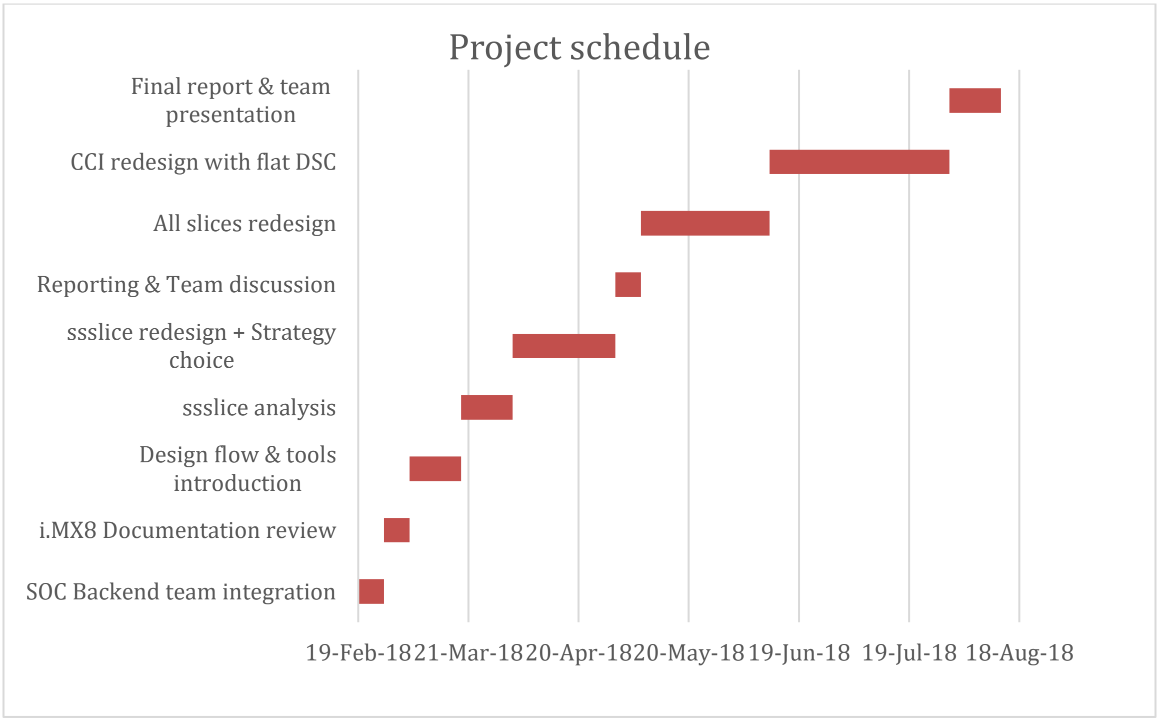


*Table 1: Project schedule*

| Steps | Start | Duration (Days) |
|---|---|---|
| SOC Backend team integration | 19-Feb-18 | 7 |
| i.MX8 Documentation review | 26-Feb-18 | 7 |
| Design flow & tools introduction | 5-Mar-18 | 14 |
| ssslice analysis | 19-Mar-18 | 14 |
| ssslice redesign + Strategy choice | 2-Apr-18 | 28 |
| Reporting & Team discussion | 30-Apr-18 | 7 |
| All slices redesign | 7-May-18 | 35 |
| CCI redesign with flat DSC | 11-Jun-18 | 49 |
| Final report & team presentation | 30-Jul-18 | 14 |

Note: There is no cost evaluation included in the report. This stems from the fact that the project didn't require any physical material. The only material used are the computer software used for design and checks. So, the cost can be related to the licenses used. However, it is irrelevant to use the licenses price to evaluate the cost as these licenses are already used by all the engineers of design teams in different sites.

## 2. DSC architecture

As previously stated, the DSC acts as an interface between the system controller unit (SCU) and each individual subsystem. It provides clocks, power control and some additional functions which may be managed by firmware running on the system controller function which in turn interfaces to software. [4]

As **Figure 2** shows, the DSC consists of two main blocks entitled dsc_dig_aon and dsc_dig_pg. The dsc_dig_aon represents the always-on part (AON), which is always supplied with power. It is implemented to ensure that the SCU keeps its control over all the DSCs of the different subsystems, even if one of the DSCs is switched off with its subsystem shut-down. This is required as the connection between the DSC and the SCU is implemented as a ring of MSI (Medium Speed Interconnect) bus showed on **Figure 3**, thus this ring needs to always function.

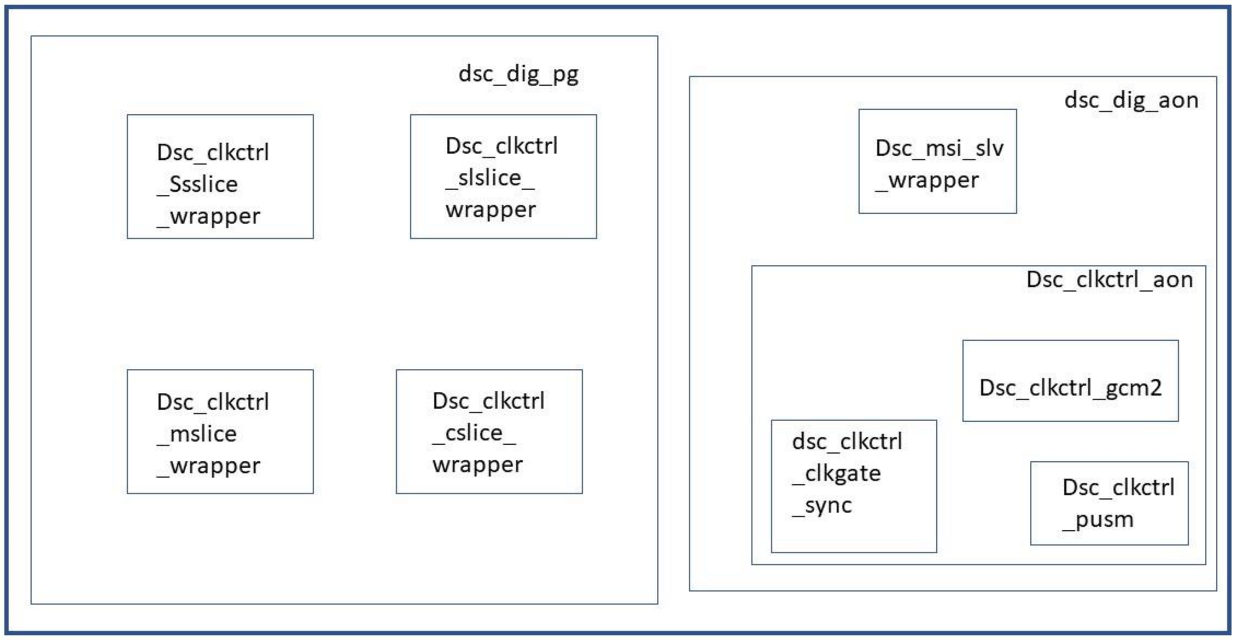

dsc_dig_pg
Dsc_clkctrl _Ssslice _wrapper
Dsc_clkctrl _slslice_ wrapper
Dsc_clkctrl _mslice _wrapper
Dsc_clkctrl _cslice_ wrapper
dsc_dig_aon
Dsc_msi_slv _wrapper
Dsc_clkctrl_aon
Dsc_clkctrl_gcm2
dsc_clkctrl _clkgate _sync
Dsc_clkctrl _pusm

*Figure 2: DSC hierarchical view of the main modules*

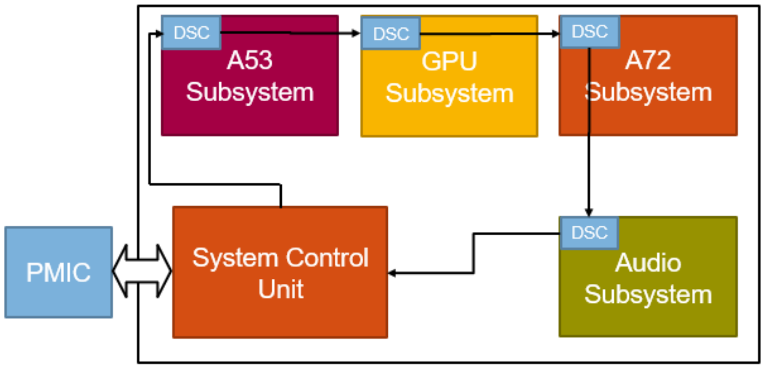


*Figure 3: MSI bus ring [3]*

The dsc_dig_pg consists of the power gated part, which can be switched off when needed. It is the biggest part of the DSC. It is responsible of the clock management, which is the main function of the DSC. It contains the clock dividers responsible for generating the required frequencies to the subsystem according to some input control signals translating the timing requirements of the subsystems. These dividers are entitled ssslice for single short slice, slslice for single long slice, mslice for multiple slice and the cslice for cpu slice. The cslice is providing clocks to the cpu and is different and is used only by one subsystem. The other slices contain the same modules and differ only by the number of instantiations of these modules. The slslice provides higher division rate than the ssslice and the mslice contains internal multiple dividers and is then capable to produce multiple clocks synchronous to each other.

## 3. Single small asynchronous clock divider slice (ssslice)

After analyzing the global architecture of the dsc, it was decided to focus on one clock divider module. This stems from the fact that the different dividers are the main high-level modules of the power gated domain of the dsc. All the dividers (except the cslice) consist of the instantiation of the same module which represent the core of the dividers. They differ only by the value of the instantiation parameters specifying the number of dividers included, the width of the chopper and the highest division value possible, which makes analyzing the architecture of one divider enough to understand how they work. At the same time, it can be expected that optimizations done on the ssslice will also be beneficial for the other dividers. Under these facts, the ssslice has been chosen as it is smaller and so it is more practical to be run with the different flows. So, it is good to work on as a starting point.

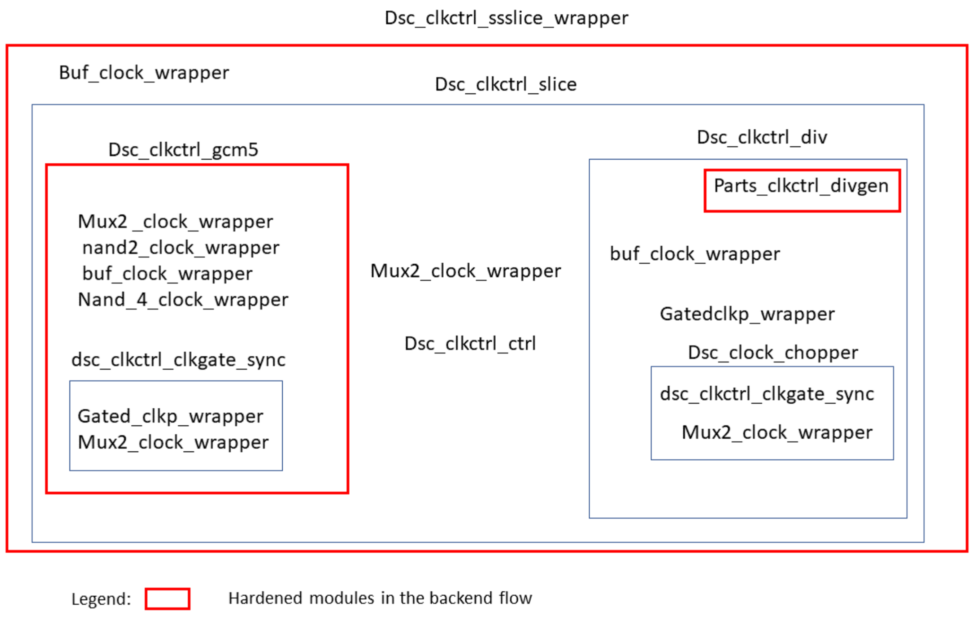


*Figure 4: Hierarchical representation of the ssslice module. It represents all the included RTL modules*

### 3.1. RTL Analysis

The RTL codes of the different modules consisting the ssslice were analyzed to understand the functional behavior of the module and to understand its architecture. This is a key point to understand the current design. The hierarchy of these modules is presented on **Figure 4** and the main ones are presented below.

### 3.1.1. dsc_clkctrl_gcm5

This module has the function of a 5-to-1 glitch-less multiplexer. It allows a safe switching between the 5 input clocks of the dsc, which are 3 pll clocks, the 24 MHz crystal clock (xtal24m_clk) and the bypass reference clock (byp_ref_clk). However, the module is designed in a generic way with respect to the clocks, which means no special considerations in the design were made for one of the clocks. It allows also the selection of the reset or the scan clock to flow in the design when needed. The clock is muxed at the input of the module with every other clock. It is one of the hardened modules inside the ssslice. As **Figure 5** shows, the module transforms the 3-bits input selection signal (src_sel), which is first synchronized through 3 dff, into a one-hot code to select one of the five input clocks. According to this code, a logic block turns on only one selection signal which is synchronized to the clock to be selected through 2 dff (same for the turned off selection signals). The Nands block, having as input the input clocks and the internal selection signals generates the right output clock. This block is the main reason of the hardening of the module, as during the design, it was found that physically distancing these Nand gate induces a duty cycle degradation. As a result, a design decision was made to physically approach the gates and then to harden the module to deal with this point only once during the backend design of the DSC.[1] A clock gate is implemented in the module to gate off clk_0 when the reset is asserted, because otherwise the reset of the synchronization dff forces the selection of clk_0 at the 5:1 multiplexer.

[1] This information was confirmed by the frontend designer of the DSC during a call.

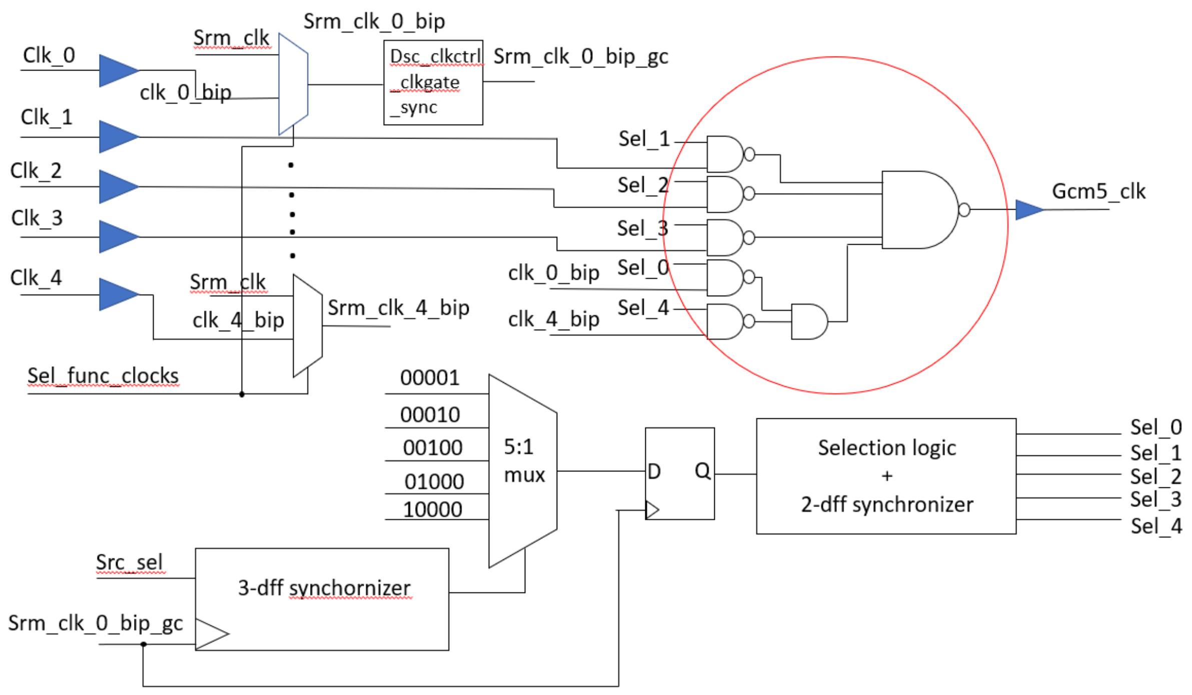


*Figure 5: dsc_clkctrl_gcm5 module schematic (All the clocks are multiplexed with the srm_clk (scan/reset clock), but for the visibility they are replaced with black dots)*

### 3.1.2. Dsc_clkctrl_ctrl

This module manages all the controlling signals of the divider. It transforms the firmware, hardware and software enabling signals coming to the ssslice to **Dsc_clkctrl_div** as one “enable” signal respecting the specifications of these signals. In addition, it checks if the divider has already an ongoing division operation before passing the new synchronized division value to the divider. It also synchronizes all the signals to the selected clock to make sure no glitches are generated.

### 3.1.3. Dsc_clkctrl_div

This module implements the actual divided clock generator **Parts_clkctrl_divgen**. It contains also the logic that feeds it according to the division factor and the enable signals. Through this logic, it allows the divider to count the rising edges (and the falling edges in the case of an odd division factor) of the input clock. It makes sure the division factor change is only taken in account when the divider finishes any ongoing division operation. The module has also a clock gate that turns off the output clock when the enable signal coming from **Dsc_clkctrl_ctrl** is 0.

### 3.1.4. Parts_clkctrl_divgen

It is the second hardened module in the ssslice. Its function is the generation of the output divided clock. Using a rising edge DFF and a falling edge DFF whose outputs are xor-ed using 2-input nand gates (to minimize the skew), it generates a 50% duty cycle divided clock. These DFF are clocked using the selected clock at the output of the dsc_clkctrl_gcm5. Their inputs are controlled by the logic in **Dsc_clkctrl_div** so that n_pedge_toggle_dat toggles for any division factor, but n_nedge_toggle_dat toggles only for an odd one (to generates the 50% duty cycle), otherwise it remains 0. As for the dsc_clkctrl_gcm5, the module is hardened to physically approach the Nand gates forming the XOR gate and the division flip-flops, which are circled in red on **Figure 6**, in order to protect the duty cycle of the clock from degradation (same as in dsc_clkctrl_gcm5).

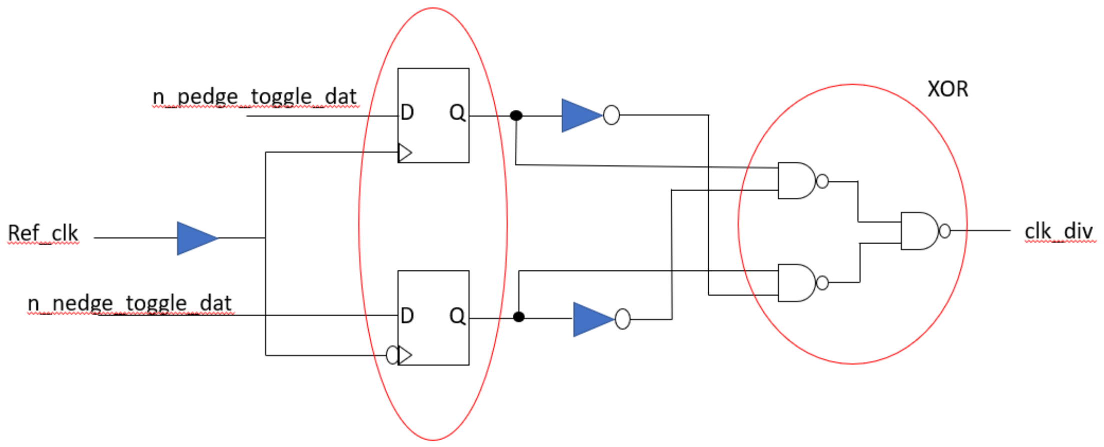


*Figure 6: parts_clkctrl_divgen schematic*

## 3.2. Performance assessment

The current performances of the ssslice were evaluated to have a reference point for the following optimization steps of the project. Some parameters were defined to be the evaluation parameters: latency of the divider between the input clock and the output (the main inconvenient of the actual dsc), the duty cycle degradation, the power consumption and the cells distribution. The goal is to optimize the latency without negatively affecting the power consumption nor the area. To analyze these parameters, a STA (static timing analysis) was run using cadence tempus tool at the post-route stage. (An STA is computation method which “determines worst case arrival time of signals at all pins of design elements”, without test functionality [5]). Seven different design corners were run, but only the worst results are reported below. Design corners takes in account PVT

(Process, Voltage, Temperature) variations. Each corner models the cells timing values and power consumption differently depending on the environment characteristics. Their use allows to have a better estimation of the circuit behavior in the design stage.

*Table 2: Latency analysis results (in ns)*[2]

| Clock | Byp_ref | Pll_0 | Pll_1 | Pll_2 | xtal24m |
|---|---|---|---|---|---|
| Latency | 0.575 | 0.537 | 0.551 | 0.529 | 0.574 |

Concerning the latency, the worst results were obtained for the corner (worst case: 0.9V, -40 °C, cmax). The latency was computed as the propagation time of a positive edge from the clock input ports to the output clock port (ssslice_clkdiv). The latency analysis required defining an output delay on a data output port, which led the tool to report the latency as the clock insertion delay in the timing report concerning this data port.

To measure the duty cycle degradation, the difference between the highest rise time and the lowest fall time as well as the lowest rise time and the highest fall time. The highest number represents the worst duty cycle degradation and was represented as a percentage of the clock period. The worst results, obtained for the corner (worst case: 0.9V, -40 °C, cmax), were 13.39% for pll_0 clock and 14.04% for byp_ref clock. These values were measured at the output clock port of the ssslice.

For the power analysis, the default "report_power" command of tempus was used. The worst results were obtained for the corner (best case: 1.1V, 125 °C, rcmax). The analysis was done with the default tempus parameters which won't change later to have a fixed reference for the power analysis. The most relevant data is the leakage power as it comes from an accurate calculation related to the cells characteristics.

*Table 3: Power consumption results (in mW)*

| Internal power | Switching power | Leakage power | Total power |
|---|---|---|---|
| 0.437 | 0.3742 | 0.2129 | 1.024 |

[2] All shown results are obtained for nominal timing constraints that are also used for constraints (see Table 5).

Table 4: Cells distribution

|  | P0 | P4 | P10 | P16 |
|---|---|---|---|---|
| Clock | 74 | 22 | 0 | 5 |
| Not clock | 68 | 74 | 42 | 295 |
| Total | 142 | 96 | 42 | 300 |

Through a script, the cells distribution was deduced from the post-route netlist. The result on **Table 4** divide the cells into clock and not clock cells to have the correlation between this distribution and the timing performance of the design. This distribution invokes also the type of the cell (P0, P4, P10, P16). Each cell in the technology library has four types. All have the same sizes, but the decrease of the figure following the P means an increase of the cell speed but also of its power consumption. It relates physically to the change of the threshold voltage Vt of the transistors in the cell (decrease of Vt => faster cell). Swapping between the different "P" is then efficient to easily enhancing the design performances or fixing timing problems. This table is reported as moving to high P values is better, so it can be used to compare the designs.

## 3.3. Physical views

As the main remark made on the dsc is the efficiency of the hardening of certain modules inside the block, a physical view of the ssslice was retrieved from innovus cadence tool (**Figure 8)** to investigate this remark. The **Figure 7** shows the physical view of the dsc_clkctrl_gcm5 module being the biggest hard marco in the ssslice. On both figures, the path of the clock pll_0, through all the nets from the input to the output, is highlighted.

As a conclusion of the analysis, the clock path in the ssslice and in the dsc_clkctrl_gcm5 is too complex. The latency being the main issue of the current dsc, this path should be optimized. This advantages flattening the ssslice to try to optimize the path in its globality to avoid adding complexity by internally hardening modules. This is supported by the fact that the hardened modules consist of z_cells[3], which means they are all conserved through the backend flow. In addition, the physical views don't show any

[3] Z_cells are technology library cells which are directly instantiated in the RTL codes and which should be kept during the whole backend flow. Their instances names start with the prefix z_cell.

special steps made for the place and route. As a result, designing the ssslice without any hard-macros will help the place and route (PnR) tool to better place the cells to optimize the latency. This should be done while keeping the gates mentioned on chapters **3.1.1** and **3.1.4** physically close to protect the duty cycle from degradation.

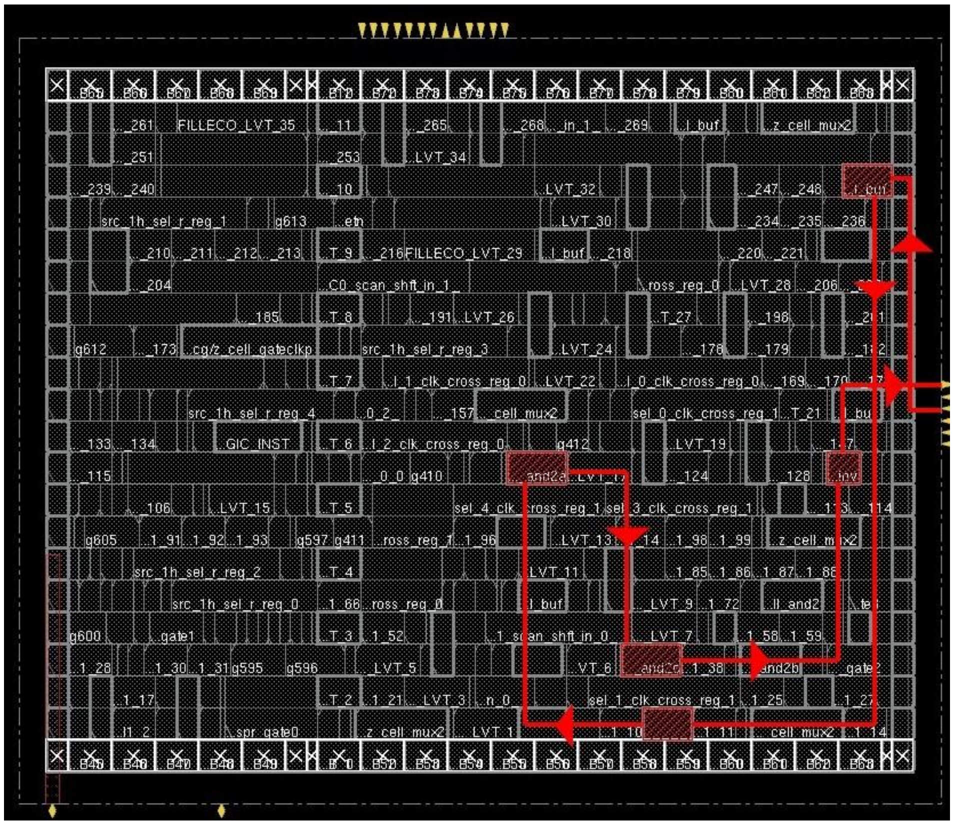


*Figure 7: Physical view of the dsc_clkctrl_gcm5 module showing the path of the clock pll_0.*

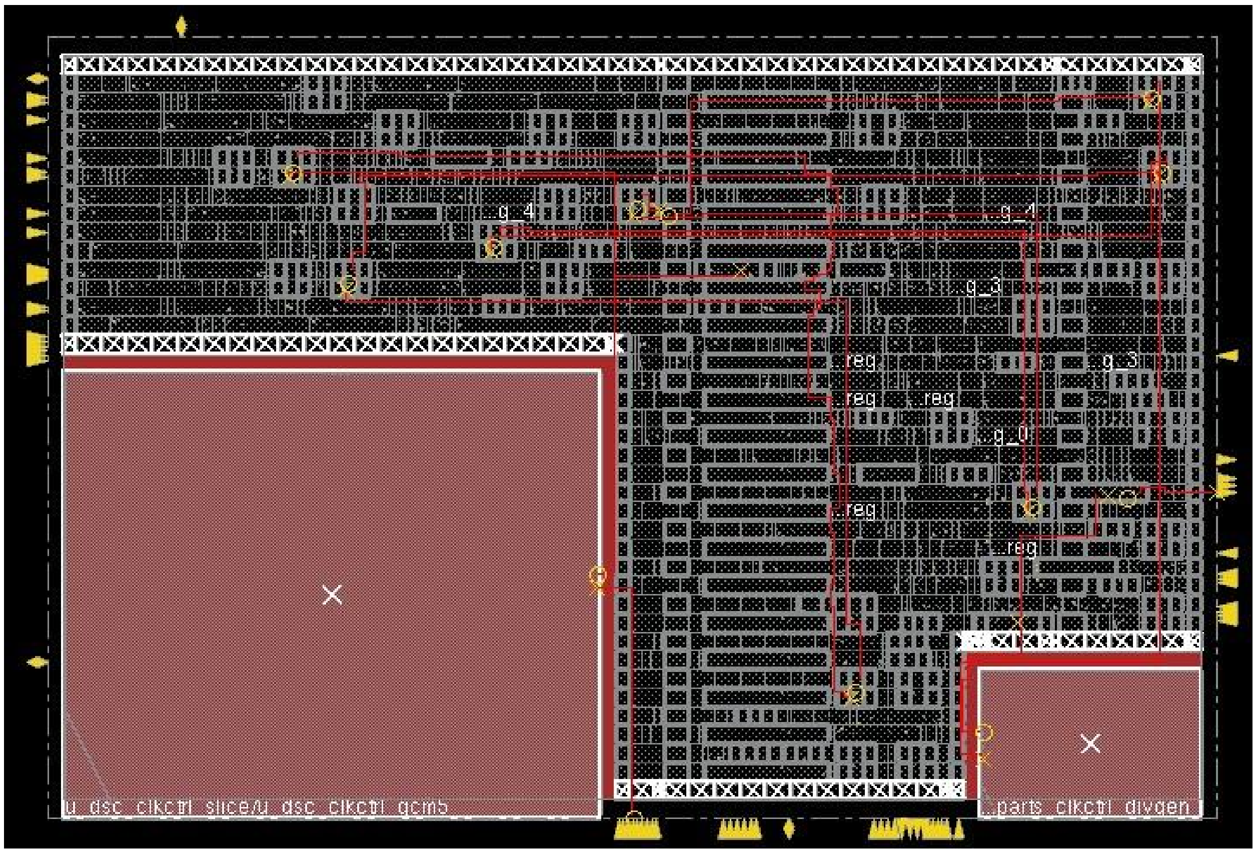


*Figure 8: Physical view of the ssslice module showing the path of the clock pll_0. (A circle is the start point of a signal and a cross is its endpoint).*

# 4. SSSlice synthesis

## 4.1. Synthesis characteristics

Synthesis is the design step related to transforming the RTL codes into a gate-level Netlist, which is a list of logic gates existing in the foundry library.

The synthesis strategy adopted as previously described is to flatten the design. All the RTL modules hierarchically present under the ssslice are fed to the synthesis tool as an input. For most of the other parameters, they were kept as they were in the current design to be able to assess the effect of flattening versus hardening.

The synthesis was performed using the 28 nm FDSOI technology of Samsung, which is the technology used for the i.MX8QM project. The used tool is the cadence tool rtl compiler. Following the current design, LVT transistors, which stands for low VT. were used. As shown on **Figure 9**, these transistors have flipped wells, as NMOS transistors have an N-well, while PMOS transistors have a P-well, and the both wells are tied to the reference voltage node VSS. These transistors are faster with the cost of the increase of the leakage power. Moreover, the synthesis concerned only one design corner, which is the worst design corner in terms of latency (see **section 3.2**).

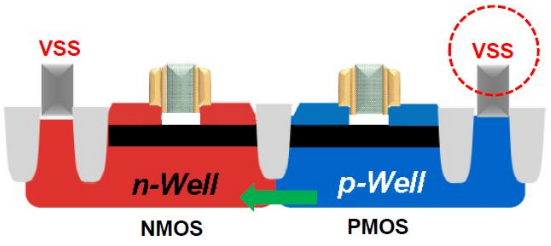


*Figure 9: LVT transistors [6]*

To guide the synthesis flow, clocks were defined as shown on Table 5. Six master clocks were defined on the input ports of the ssslice. Among them, there are five functional clocks and the scan/reset clock. From these clocks, a first series of generated clocks are defined on the output pin of the Nand gate at the output of parts_clkctrl_divgen (see **Figure 6**). These clocks represent the divided clocks generated from these master clocks. To avoid overloading the timing constraints file with clocks definitions for every division rate and for every clock, these clocks were defined as divided by one. The division by one is behaviorally allowed in the DSC design and is then the most critical case from timing point of view. From these generated clocks, another series of generated clocks were defined as divided by one at the output clock port of the ssslice module. A generated clock from the scan/reset clock was also defined on the same point.

The synthesis flow handled also the scan chain insertion. For this, all the flip-flops in the design were forced to be scan flip-flops to make the process easier. The insertion of multi-bit flops was allowed in the flow, which means the tool uses multi-bit cells instead of separate flip-flops when it finds optimal for the design. Multi-bit structures consist of flip-flops combined into one cell, which makes them smaller. The logic values of the different scan signals were defined for the scan mode to allow the tool to insert the chain. Only one chain was inserted in the design. This differs from the actual design were 2 scan chains exist, however, the insertion of 2 scan chains was only based on the maximum number of flip-flops per chain allowed by the design flow of the backend team who designed the module. Therefore, only one scan chain was inserted in this new design.

*Table 5: Defined clocks*

| Clocks | Frequency (MHZ) | Master/Generated |
|---|---|---|
| dsc_FCK_PLL_0_1p2G | 1316 | Master |
| dsc_FCK_PLL_1_1p2G | 1316 | Master |
| dsc_FCK_PLL_2_1p2G | 1316 | Master |
| dsc_FCK_BYP_REF_CLK | 1316 | Master |
| dsc_FCK_XTAL_24M | 100 | Master |
| dsc_FCK_SRM_100M | 100 | Master |
| dsc_FCK_G1_SSSLICE_DIV_PLL_0 | 1316 | Generated |
| dsc_FCK_G1_SSSLICE_DIV_PLL_1 | 1316 | Generated |
| dsc_FCK_G1_SSSLICE_DIV_PLL_2 | 1316 | Generated |
| dsc_FCK_G1_SSSLICE_DIV_BYP_REF_CLK | 1316 | Generated |
| dsc_FCK_G1_SSSLICE_XTAL_24M | 100 | Generated |
| dsc_FCK_SSSLICE_DIV_PLL_0_out | 1316 | Generated |
| dsc_FCK_SSSLICE_DIV_PLL_1_out | 1316 | Generated |
| dsc_FCK_SSSLICE_DIV_PLL_2_out | 1316 | Generated |
| dsc_FCK_SSSLICE_DIV_BYP_REF_CLK_out | 1316 | Generated |
| dsc_FCK_SSSLICE_XTAL_24M_out | 100 | Generated |
| FCK_SRM_SSSLICE_100M_out | 100 | Generated |

## 4.2. Aftermath checks

Two checks were run on the gate level netlist resulting from the synthesis step: STA and LEC. These checks are systematic checks dictated by the design flow used by the team.

The STA at the post-synthesis stage aimed to check for setup timing violations, which are the relevant violations to be fixed at this stage. The results of STA showed few violations coming from the asynchronous reset port (aresetn) and from the scan enable port (scan_en_dsc). However, these violations are irrelevant violations at this stage since these signals should be routed everywhere in the design to drive all the flip-flops. These signals need then to be buffered and balanced like the clock signals. But this is not considered by the STA at this stage. These violations are then masked by adding two timing constrains to set the paths coming these ports as false paths. These constraints were only considered for the STA at the post-synthesis stage.

The LEC (Logic Equivalence Check: it checks if two Verilog files are logically equivalent, ex: RTL vs Netlist or Netlist vs Netlist) was run to verify that the Netlist is logically equivalent to the RTL code. The check result shows only one non-equivalence at the output port of the scan-chain. This non-equivalence was already expected as on the RTL code, this output port is floating but is tied to a scan-flop output on the netlist. The LEC was then identified as clean. It proved the synthesis and the scan insertion were done correctly.

# 5. SSSlice Place and Route

The Place and route step consists of transforming a gate-level netlist into physical layout, reaching the GDSII fil which can then be sent to the foundry to manufacture the circuit.

## 5.1. Design characteristics

The design was made with the cadence tool innovus version 16.24.00. It was divided into 4 main steps: init, place, CTS and route.

The init step is the initialization step. It included adjusting the design settings and setting the floorplan, which is the physical architecture of the block and in which the cells are placed. The block size is kept as the current design 50.32*33.6 $\mu m^2$. The actual boundary of the cells placement is distanced from the edges of 1.6 µm. The placement of the I/O pins is also kept like the current ssslice to be able to have a relevant comparison. For this, the I/O placement was extracted on Innovus, from the current design, as a tcl script which is sourced in the flow. Some physical only cells are added in the floorplan. They don't have logic functionality but are necessary for the well-functioning of the circuit. Among these cells, decoupling capacitances are placed all around the block. They "act as a capacitance between power and ground rails, and hence as a charge reservoir that can be counted upon while there is a high demand for current from the power lines" [7]. The other kind of physical only cells are well-taps. They are required for "tap-less" libraries, where not every cell has these in-built tap connections, but are smaller. The well-tap cells connect the N-wells and the substrate to VDD/GND [7]. Also, in this step, power switches and vias are placed. Power switches are required for switching on/off power gated domains. The ssslice is totally present on a power gated domain, however the current design has an always-on power domain needed for some power switches that require a continuous supply. This kind of switches is no longer used, so the always-on domain is removed in the new design. The inout power ports are placed and routed to create the power nets. This routing is done on metal layers 5 and 6, as the use of metal should be limited in all the design to metal 6 as the design deals with a block. This aims to make the block integration easier in the subsystem (until M8 allowed) and consecutively in the top-level architecture of all the chip (all metal levels).

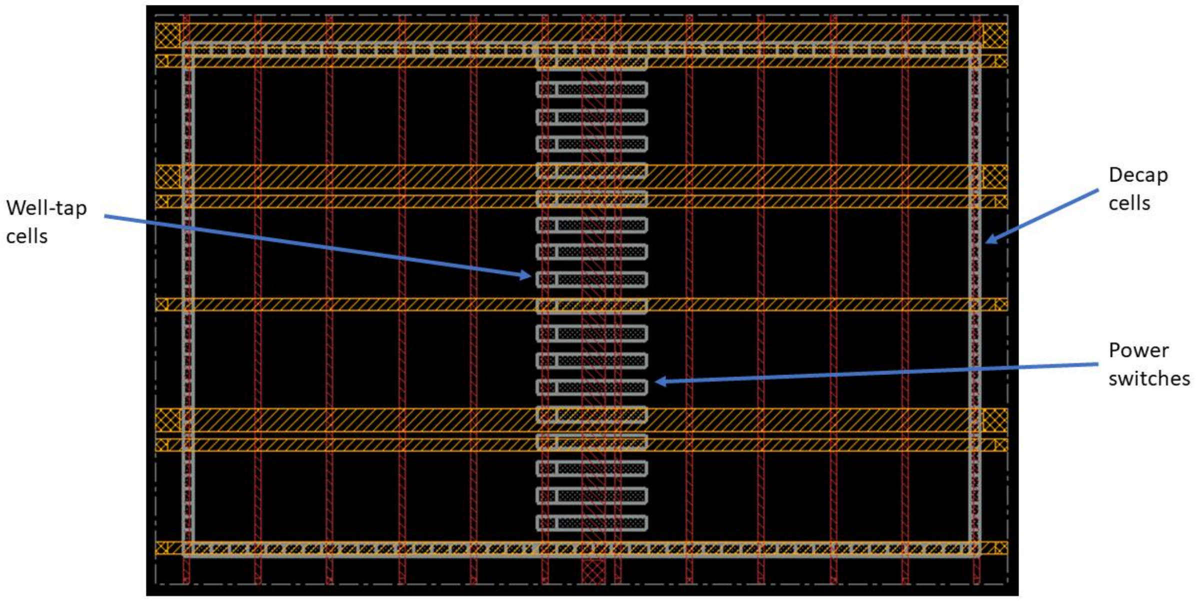


*Figure 10: ssslice floorplan*

The place step handles the cells placement on the designed floorplan. This is done with the aim of fixing timing violations (setup, hold, transitions, …) and optimizing the power consumption. Cells can also be swapped with others from the technology library. Only two constraints were applied to this step and took effect also for the next ones. As discussed on **section 3.1.1** and on **section 3.1.4**, some logic gates should be physically close on the design to avoid clocks duty cycle degradation. One solution was to implement fences in the design. They are physical boundaries, in a specific location in the design, in which the tool is constrained to add the cells of the specified gates. This requires choosing the right location for these fences to optimize the latency and this can require many design loops to find the positions. Therefore, another solution was chosen to apply the placement constraints. The command "create_inst_space_group" tells the tool to spatially approach the cells corresponding to the instances specified on the command line, without giving an area limit or a specified location. This gives the tool more liberty to move the cells, while respecting the constraints, and therefore it allows the tool to optimize the location to optimize the latency. The result can be seen on **Figure 11**, where the cells of parts_clk_divgen are highlighted in green and those of dsc_clkctrl_gcm5 in red. It shows the success of the command in executing the required job, as the cells are kept close.

The CTS (clock tree synthesis) handles the insertion of the clock tree. As a reminder, the goal of a clock tree insertion is to buffer the different clocks to the different point in the design, while minimizing as much as possible the skew, which is the difference between the time of arrival of a clock to two different points. The CTS tool implemented in innovus bases the insertion on the clocks definitions in the timing constraints file. It divides the mission in two steps. Starting from an input clock port, it starts by buffering the clock to the different arrival point in the design. This step is called clustering. Then to minimize the skew between the different arrival points of the clocks and to have more homogenous arrival times, it adds few buffering stages on some branches. This step is called balancing. Additional constraints can be added to help the tool, for instance to help it to identifying the flip-flops, used for clock division, and which should be integrated in the clock tree branches. A constraint was added to stop the scan/reset clock (srm_clk) where it is muxed with functional clocks. This means it is only buffered to these points. This clock being a slow clock used for scan shift purposes, this constraint avoids having long clock tree branches for it. A similar constraint was also added for the clock xtal24M_clk (see **Table 5**). This clock is also a slow clock that could penalize other clocks by leading to longer branches. The tool is constrained also to use inverters to synthesize the clock tree. As cells have different propagation times for rise and fall, using buffers can lead to a duty cycle degradation for long branches, but using inverters continuously swap rising edges with falling edges and vice-versa, which helps countering the problem. The tool is also guided to use only P4 cells as they are a good compromise between speed and power consumption, taking in account that speed is more important for CTS. Once the whole tree is placed, the clock cells are marked as fixed, so the tool doesn't try to change their placement lately. This is a protection constraint, as the CTS is a hard step that once is done should not be changed a lot.

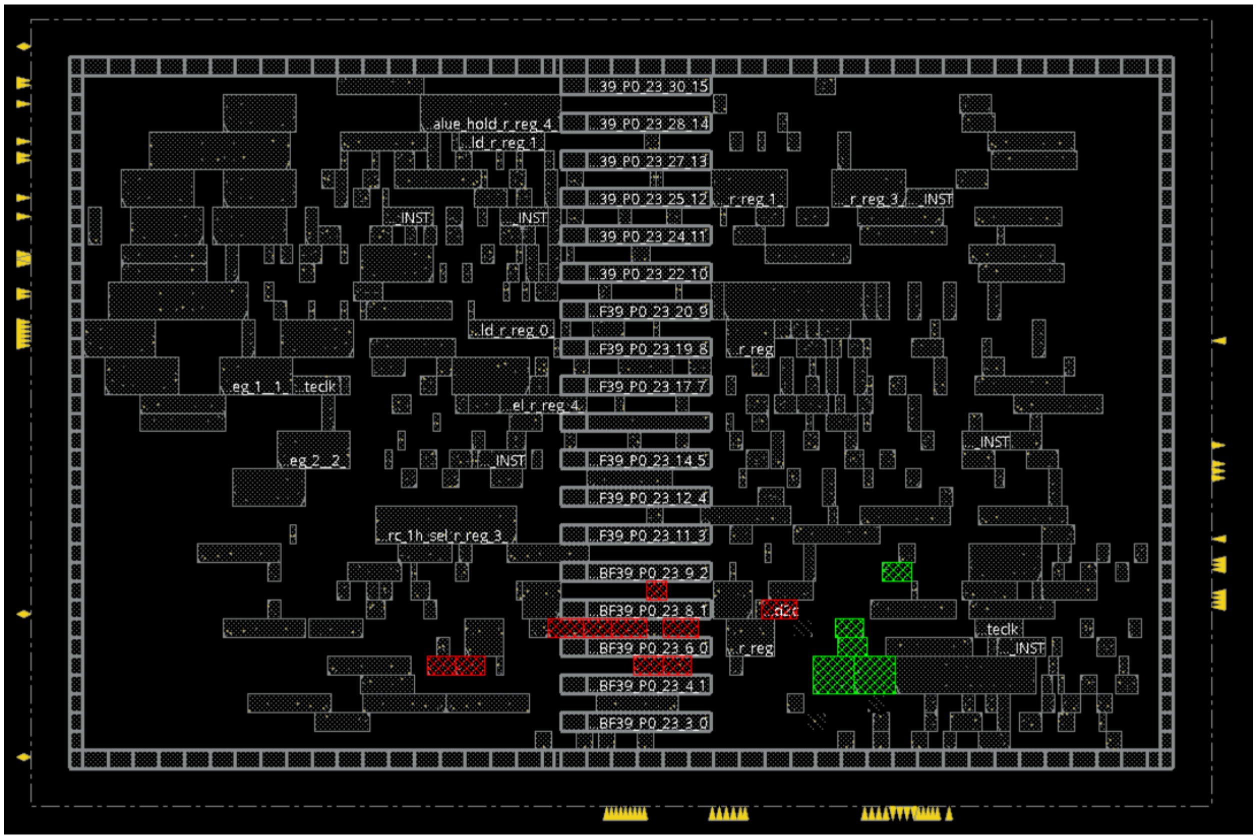


*Figure 11: ssslice after place step*

The route step handles routing all the nets, but also the input and output ports. As previously stated, it can use the different layer of metals without going higher than metal 6. The tool still tries in this step to fix timing violations and to reduce parasitic (that can lead to timing violations as well). It adapts the routing for the purpose and can swap some cells with equivalent ones from the technology library or just change their placement. However, as the CTS step is the most complex, once the clock tree cells are placed and optimized, they are fixed in their place and not touched anymore by the tool.  The tool fills also all the remaining gaps in the design, after this step, with fillers, which are physical only cells. They are essential to have a clean DRC (for DRC, see **section 5.2**). They "continue the base layers like NWELL and have the VDD/VSS pins matching the rest of the standard cells" to avoid NWELL/power connection continuity problems [8] . **Figure 12** shows the final layout obtained.

At the end of the PnR step, some important files are extracted. The gds files is a data base binary file, which can be sent to the foundry for the manufacturing step. It contains all the necessary information concerning hierarchal forms and the geometrical shapes.

The lef files contains the physical information and the design rules of the cells in the design. Their representation of the physical layout is completed by the def files.

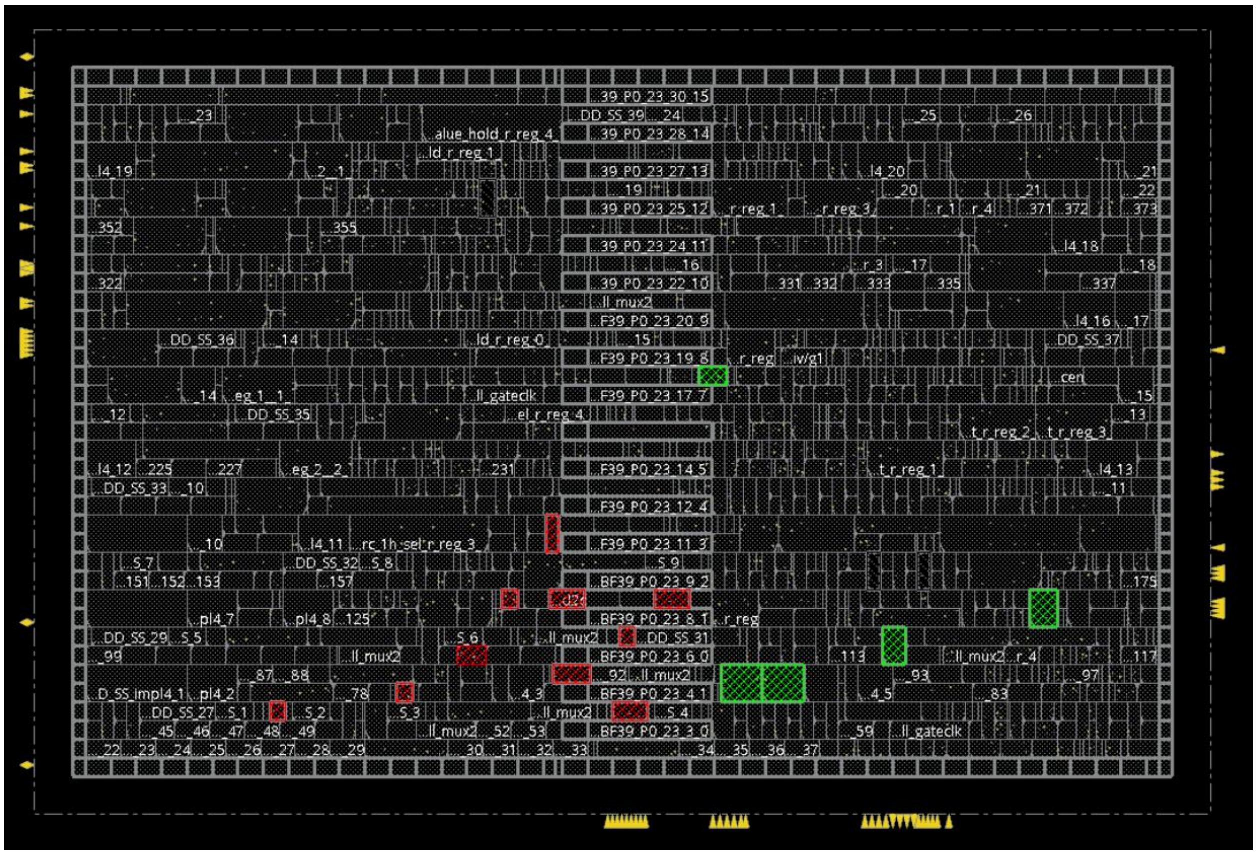


*Figure 12: ssslice final layout*

## 5.2. Post-layout checks

After the design was completed, physical checks were run to verify that the design comply with certain criteria. These checks were run through Calibre (a mentor graphics tool). The essential ones are DRC and LVS checks. The DRC (Design Rule Check) verifies that the design respects some semiconductors manufacturing parameters, which ensure that the correctness of the mask and then the physical design. These parameters define mainly geometrical requirements. For the ssslice, the check resulted in one error. However, according to the tool manual explanation of the error, it is due to a command used in the route step and it should be corrected by the command "ecoRoute" used in the eco step (see **section 5.3**). So, it is left to be corrected in this step. On the other hand, the LVS (Layout Versus Schematic) verifies that the generated post-layout netlist, which is used for the following checks, corresponds to the designed layout. It mainly verifies both have the same internal connections. The check shows no violations.

After this step, the parasitic models of the design are extracted in spef files format. Spef (Standard Parasitic Exchange Format) files describe the resistance and capacitance parasitics of all the nets in the design. They are an essential input for the STA as they allow it to have an image of the design less ideal and closer to the reality.

*Table 6: Clocks latencies for flattened ssslice (ns)*

| Clock | Byp_ref | Pll_0 | Pll_1 | Pll_2 | xtal24m |
|---|---|---|---|---|---|
| Latency | 0.436 | 0.415 | 0.404 | 0.416 | 0.451 |

*Table 7: Power consumption for the flattened ssslice (mW)*

| Internal power | Switching power | Leakage power | Total power |
|---|---|---|---|
| 0.505 | 0.459 | 0.08 | 1.044 |

The STA was run on the post-route stage using the extracted files. The results show hold and minimum pulse width (MPW) timing violations. On one hand, the MPW violations are fixed in the eco stage (see **section 5.3**). On the other hand, the hold violations were either from (to) input (output) ports (Violations that should be fixed on the higher level of hierarchy), or to test input pins (Violations that existed already in the current design and were masked). So, the corresponding paths were set as false to mask these violations. Then, the clocks latencies and the power consumption were measured with the same way as before (see **section 3.2**). The results are shown on **Table 6** and **Table 7**. The latency results show that the xtal24m clock got the higher latency in the new design. This can be explained by the clock tree synthesis privileging the other clocks over it, due its functionality. Taking then the byp_ref clock as the reference, a decrease of 24% in the latency resulted from the new design. For the power consumption, the results show an increase of 2%, which is completely acceptable as the DSC consumption, in itself, is not critical and is small in front of the latency decrease which is expected to have an overall decrease of the power in the subsystem and chip levels. However, there was still room for improvement which triggers the eco.

A LEC was run on the post-layout netlist versus the RTL. The result is like the previous one (**section 4.2**) and was identified as clean.

## 5.3. Timing ECO

Engineering Change Order (ECO) is the process of modifying the PnR netlist in order to meet timing requirements [9]. It involves swapping cells, changing cells size or cells placement, adding/deleting cells, without redoing the whole flow again, which decreases the run time.

The timing eco was triggered to fix the 2 MPW violations found in the STA results and to improve the latency. It was also required to fix the DRC violation previously found. It consisted of swapping all the cells, that are not P4, on the clocks path to P4 cells which are faster. The change lead to the new results shown on **Table 8** and

**Table** 9**.** The latency decreased more and reached 29% with respect to the current design (28 ps less than before to reach a total decrease of 167 ps), without a relevant increase of the power consumption (It remains at 2%).

However, these changes were not enough to fix the MPW violations. The two DFF, which are the endpoints of the two violating paths, had to be swapped from P16 to P4. This change was added in the same timing eco loop. The results were then clean.

The eco flow involves an eco-route which corrects the DRC violation. This was verified by running a PV check using Calibre.

At this stage, the degradation of the duty cycle was evaluated as in the analysis stage (see **section 3.2**). The results did not change with the new design: 14.04 % for pll_0 (vs 13.39% previously) and byp_ref (vs14.04% previously).

*Table 8: Latency results after timing eco (ns)*

| Clock | Byp_ref | Pll_0 | Pll_1 | Pll_2 | xtal24m |
|---|---|---|---|---|---|
| Latency | 0.408 | 0.387 | 0.380 | 0.392 | 0.426 |

*Table 9: Power consumption after timing eco (mW)*

| Internal power | Switching power | Leakage power | Total power |
|---|---|---|---|
| 0.505 | 0.459 | 0.0823 | 1.045 |

As a conclusion, flattening the ssslice reveals to be a better design strategy for the dividers than the internal hardening of dsc_clkctrl_gcm5 and parts_clkctrl_divgen. It reduces the clock latency between the input and the output by 29% with only 2% increase of power and without increasing the duty cycle degradation. The leakage decreases of 60.4%, which is huge.

# 6. Flattened DSC

Continuing with the same philosophy used to design the ssslice for the dsc level, a new design strategy was thought off. Instead of hardening the DSC in the subsystem, another solution can be to implement only the dividers and the msi_slv module (implemented on the AON part) as hard macros and to flatten the remaining of the dsc in the subsystem. These parts were already hardened in the current version of the dsc. Flattening can give the PnR tool more freedom in placing the logic cells where the latency would be optimal (as it was the case inside the ssslice), according to the clock requirements of the subsystem itself. This design strategy was adopted on the CCI[4] subsystem to evaluate its efficiency.

## 6.1. More Improved Ssslice

The new design of the ssslice, previously shown, was intended to be a proof of concept to compare the flattening versus the hardening of the modules in the dividers. This required keeping certain parameters as they were in the current design (like the die area, the I/O pins placement) to have a faire comparison. As the flattening was proofed to be more interesting, two other design strategies were exploited to further improve the ssslice latency when the module is directly implemented in the subsystem as a hard macro.

### 6.1.1. Pre-placed clock cells

As in the new design, the clock path between the input and the output is limited to z_cells instantiated in the RTL (CTS cells on the path don't have the upper hand on the latency control), these cells can be preplaced manually in the design during the init step. This allows to minimize the physical distance between them to reduce the latency. **Figure 13** shows the used preplacement, which represents an L-shap. This involved moving The IO pins as shown to comply with the placement of the cells.

[4] CCI is one of the subsystems of the i.MX8 QuadMax microcontrollers. It is responsible for make sure the two ARM cores implemented are notified of main memory changes with respect to their cache memories.

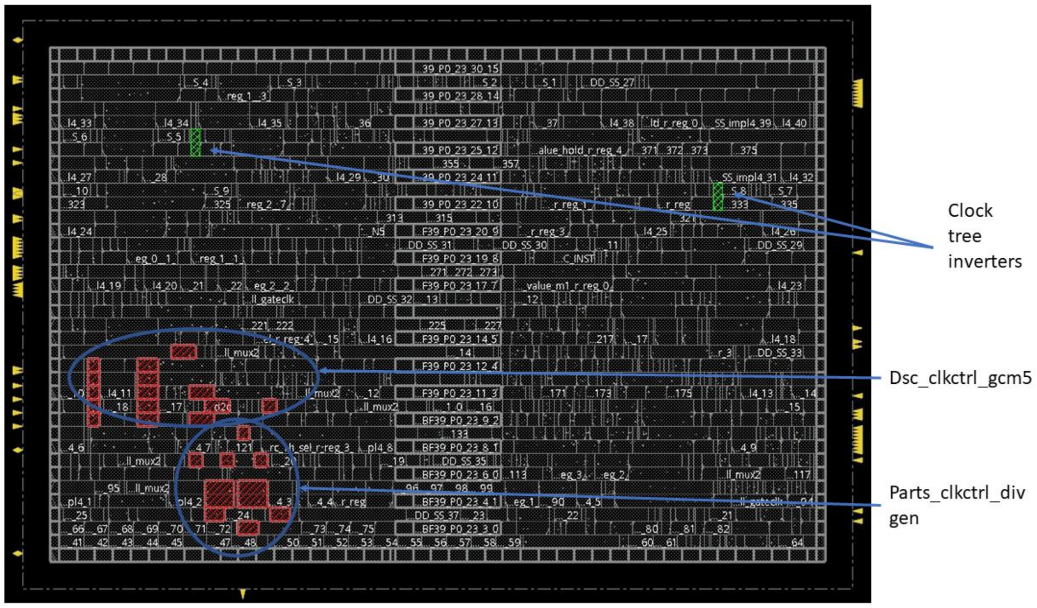


*Figure 13: Ssslice layout with preplaced z_cells*

However, this strategy didn't show improvements. The obtained latency is of 417 ns for byp_ref clock (vs 408 ns previously obtained) with a power consumption of 0.98 mW. The latter risks increasing while fixing timing violations. The first run showed many timing violations, which requires effort and time to fix without further improving the latency. This method failed to improve the latency because the CTS tool was obliged to balance the flops of parts_clkctrl_divgen with other points in the design. To do so, it placed two inverters very far physically from the preplaced cells as the distance between them is too small to have a long buffering stage reaching the flops. The strategy proofed to be inefficient and was not further developed.

### 6.1.2. Smaller Ssslice

One of the main elements affecting the clock latency is the physical distance between its input and its output pins. So, decreasing the die area of the ssslice should lead to a decrease of the latency. In the new design, the cells density with the respect to the total die area was 51.87%. This low value justifies that the area can be decreased without causing problems for the tool to adjust cells placement. Decreasing the size of the dividers is also useful for the physical implementation in the subsystem. These elements triggered another design improvement path which involved decreasing the die area and replacing the input clock pins on the left side and the output clock pin on the right. This pin placement naturally pushes the tool to place the clock cells along the straight path from the input to the output, which allows a further decrease of the latency. This modification required few iterations to find a convenient trade-off between area, latency and power. The final design, which is implemented in the CCI, is shown on **Figure 14**. Its dimensions are 45.288*30.4 which represents a decrease of 18.6% in area and a cells density of 84.5%. For this design during the CTS step, capacitances cells were added at both sides of each clock cell. They are useful to give the clock cells a private protection against parasitic effects and voltage drop.

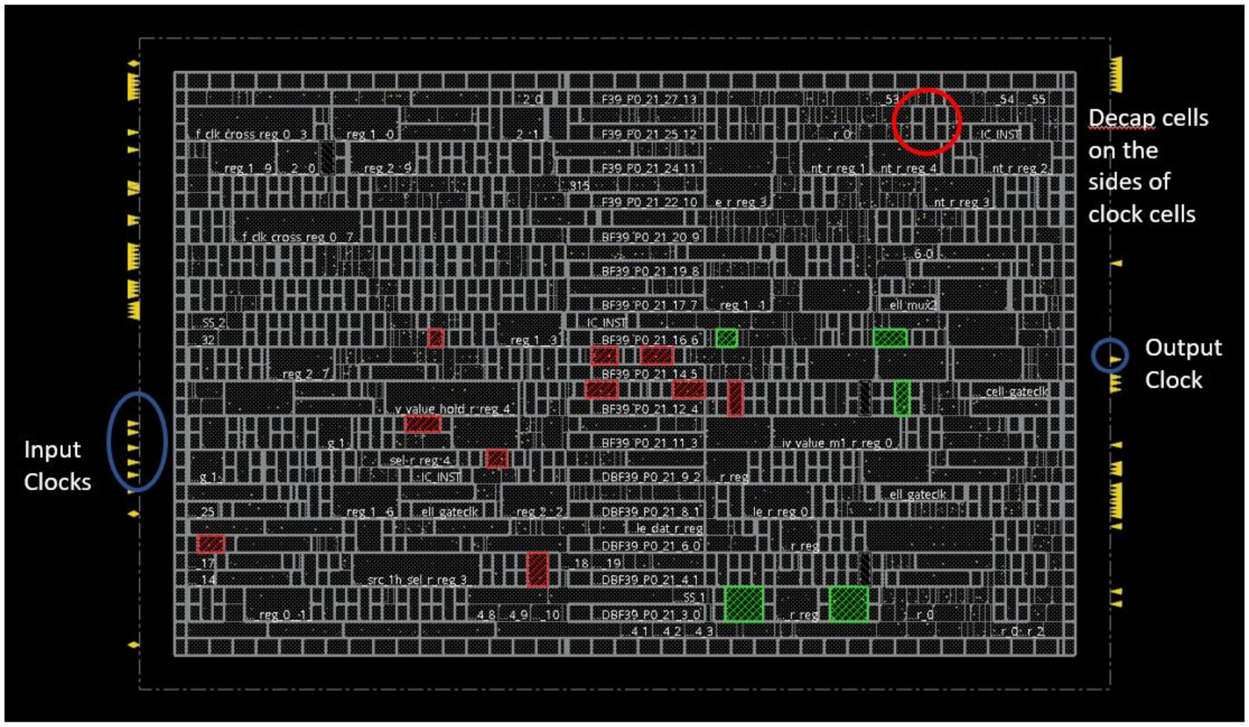


*Figure 14: ssslice final design implemented in the CCI*

For timing fixes, additional PVT corners for overdrive and underdrive conditions were taken in account in the STA to ensure a safer integration in the CCI. This required additional timing constraints which are represented on **Table 10**. The obtained performances are summarized on **Table 11** which shows a 33.2% decrease of the latency of byp_ref clock, which has the worst latency, with a smaller duty cycle degradation. It shows a 9% increase of power. However, this increase is due to the increase of internal and switching power, but leakage power decreased of 58%. The power results remain positive as the leakage power is more relevant for a global power decrease on the subsystem level. For this reason, an additional step was added to the design steps. tempus tool was used in eco mode to optimize the leakage power (to decrease it as much as possible). In this mode, tempus checks for all the possible swaps of cells that can decrease the leakage without further downgrading timing or causing violations.

*Table 10: Timing constraints with overdrive (OD) and underdrive (UD) definitions*

| Clock Name | Clock Type (Created, Virtual, Generated) | Frequency (Mhz) UD | NM | OD |
|---|---|---|---|---|
| dsc_FCK_PLL_0_1p2G | CREATED | 650 | 1316 | 1600 |
| dsc_FCK_G1_SSSLICE_DIV_PLL_0 | GENERATED | 650 | 1316 | 1600 |
| dsc_FCK_PLL_1_1p2G | CREATED | 650 | 1316 | 1600 |
| dsc_FCK_G1_SSSLICE_DIV_PLL_1 | GENERATED | 650 | 1316 | 1600 |
| dsc_FCK_PLL_2_1p2G | CREATED | 650 | 1316 | 1600 |
| dsc_FCK_G1_SSSLICE_DIV_PLL_2 | GENERATED | 650 | 1316 | 1600 |
| dsc_FCK_BYP_REF_CLK | CREATED | 650 | 1316 | 1600 |
| dsc_FCK_G1_SSSLICE_DIV_BYP_REF_CLK | GENERATED | 650 | 1316 | 1600 |
| dsc_FCK_XTAL_24M | CREATED | 24 | 24 | 24 |
| dsc_FCK_G1_SSSLICE_XTAL_24M | GENERATED | 24 | 24 | 24 |
| FCK_SRM_100M | CREATED | 50 | 100 | 100 |
| dsc_FCK_SSSLICE_XTAL_24M_out | GENERATED | 24 | 24 | 24 |
| dsc_FCK_SSSLICE_DIV_BYP_REF_CLK_out | GENERATED | 650 | 1316 | 1600 |
| dsc_FCK_SSSLICE_DIV_PLL_2_out | GENERATED | 650 | 1316 | 1600 |
| dsc_FCK_SSSLICE_DIV_PLL_1_out | GENERATED | 650 | 1316 | 1600 |
| dsc_FCK_SSSLICE_DIV_PLL_0_out | GENERATED | 650 | 1316 | 1600 |
| FCK_SRM_SSSLICE_100M_out | GENERATED | 50 | 100 | 100 |

*Table 11: Performances summary of the new improved design*[5]

| Performance | Current design | New design | Improved new design |
|---|---|---|---|
| Latency | 0.575 ns | 0.408 ns | 0.383 ns |
| Duty cycle | 14.04% | 14.04% | 12.74% |
| Power | 1.024 mW | 1.045 mW | 1.117 mW |
| Area | 1690 µm^2 | 1690 µm^2 | 1377 µm^2 |

[5] The shown results are obtained for nominal conditions and for the corners showing worst results which are the same as previously mentioned.

## 6.2. Slslice design

The slslice is an acronym for Single long slice. Like the ssslice, it generates a single output clock that can be divided with respect to its input clock. The internal structure is exactly the same. The difference is in the maximum division rate that the divider can generate. The slslice can divide by 1-255 while the ssslice can only divide by 1-31. This makes the width of the registers handling the value of the division rate larger than in the slslice, which makes its die area bigger.

Under these conditions, the slslice was designed adopting the same strategy as for the ssslice. The previously discussed modules were flattened, and the area was decreased from 1932 µm$^2$ (50.32*38.4) to 1623 µm$^2$ (46.104*35.2) with a useful cells density of 73%. The I/O pins locations were changed and the same characteristics and constrains of ssslice were adapted to the slslice case. Tempus eco was run on the design to ensure leakage power optimization is not missed. The obtained design is shown on **Figure 15.**

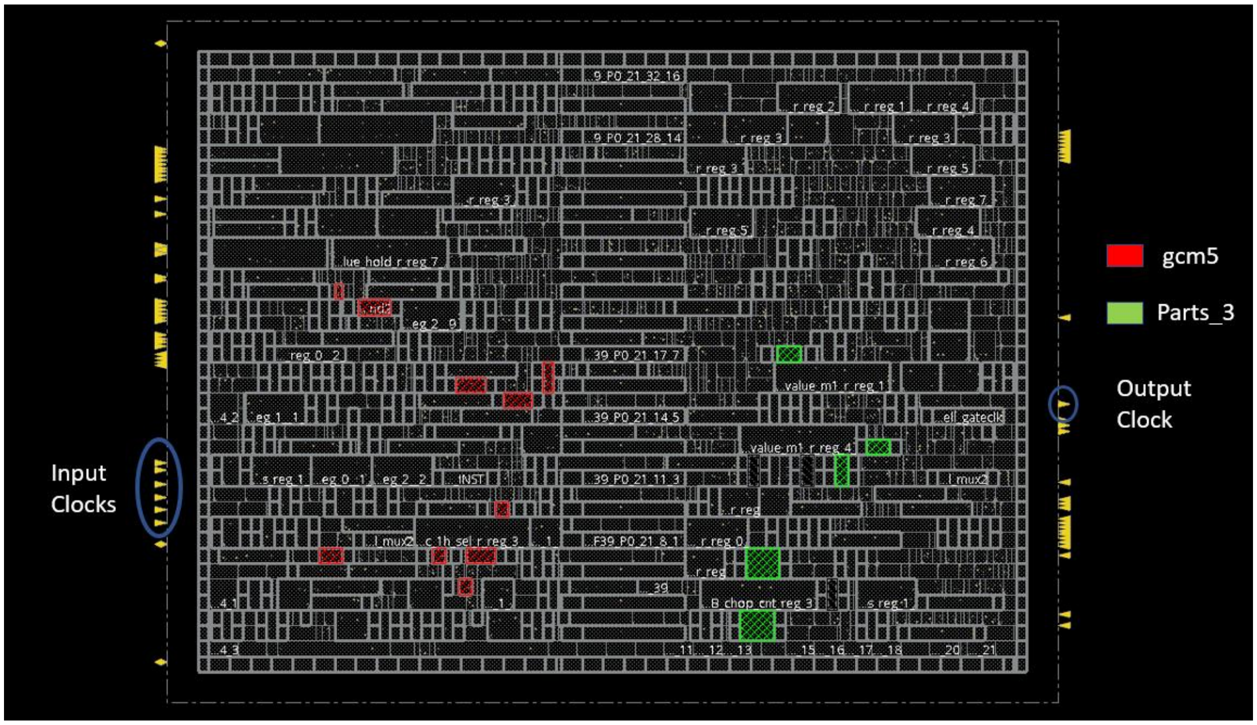


Figure 15: slslice final design

The same analysis was run through the STA on slslice and the results are summarized on **Table 12**. These results are obtained for nominal conditions and for the worst cases

PVT corners. The actual values were not evaluated to gain time; however, the new results are even better than the ssslice results.

*Table 12: Slslice analysis summary*

| Latency | | Duty Cycle degradation | | Power consumption | |
|---|---|---|---|---|---|
| byp_ref | pll_0 | byp_ref | pll_0 | Total | leakage |
| 366 ps | 342 ps | 15.99% | 15.21% | 0.636 mW | 0.0472 mW |

## 6.3. Mslice design

The mslice is the acronym for multiple slices. The module is the equivalent of 4 single short slices. It generates four output clocks synchronous to each other, even if each one of them can have different division rate. This is translated structurally by having 4 instantiations of parts_clkctrl_divgen module, which increases the die area with respect to the slslice and the ssslice. The same design strategy as for the others was adopted in designing the mslice. Additional CTS constraints were added to ensure the 4 output clocks are synchronous to each. The constraints defined a sperate skew group containing the clock pins of the toggling flip-flops contained in the 4 parts_clkctrl_divgen modules.

The design is presented on **Figure 16**. The die area decreased from 4143 µm$^2$ (76.16*54.4) to 3777 µm$^2$ (71.536*52.8) with a useful density of 63.1%. This value is not so high which means the mslice die area can be further decreased. But, as the die area decrease requires many iteration of the whole flow, the decrease was stopped at this level. The design being bigger and more complicated than the previous slices, it required using tempus eco to clean setup and hold violations. It was also used to run a leakage power optimization as for the other designs.

The latency and power consumption were evaluated using tempus by running a STA as previously done. The results are shown on **Table 13** and **Table 14**. They are close to the ones of the ssslice and the slslice taking in account the larger area. In addition, the latencies reported for each slice are close to one another. The variation between the maximum and the minimum latency is 4 ps for byp_ref and 5 ps for pll_0. This proves the synchronicity of the output clocks.

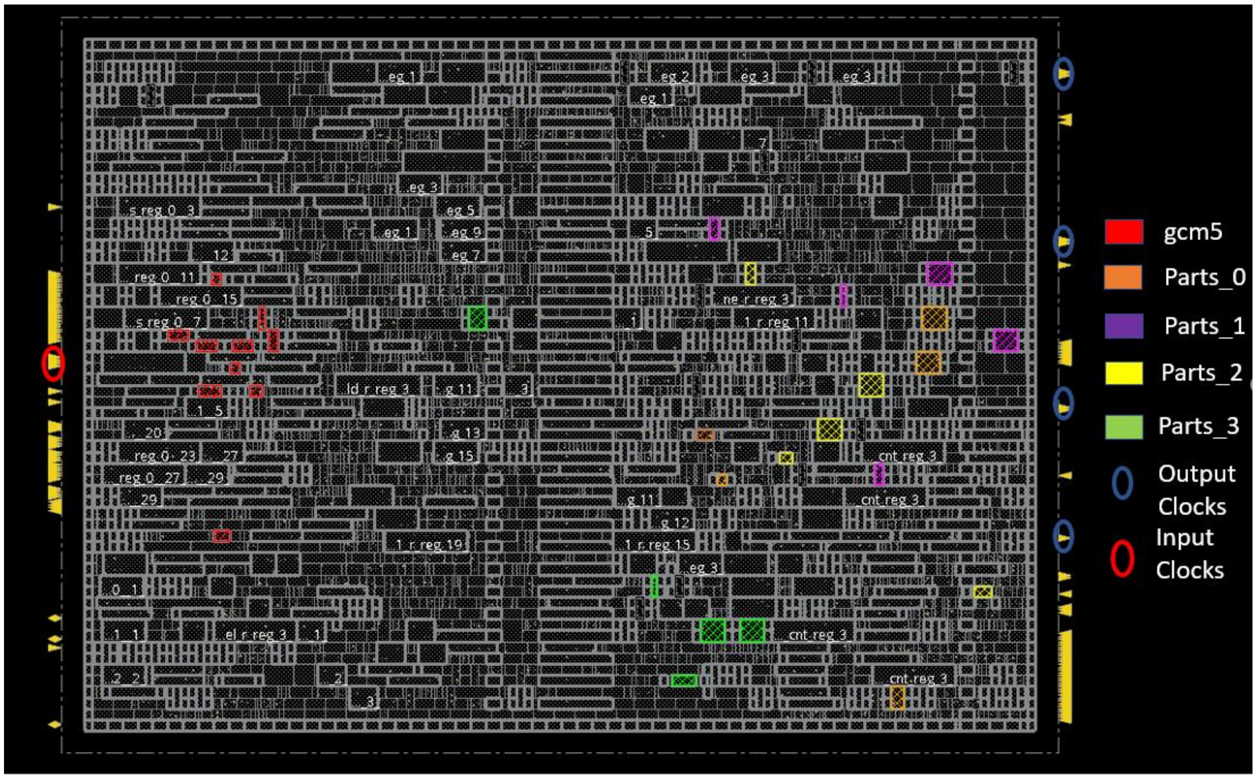


Figure 16: mslice final design

*Table 13: Mslice latency results*

| mslice # | Latency | |
|---|---|---|
| | byp_ref | pll_0 |
| mslice0 | 401 ps | 377 ps |
| mslice1 | 405 ps | 382 ps |
| mslice2 | 403 ps | 379 ps |
| mslice3 | 403 ps | 380 ps |

*Table 14: Mslice power results*

| Power consumption | |
|---|---|
| Total | leakage |
| 2.15 mW | 0.216 mW |

### 6.4. Integration in CCI

To Assess the new design strategy, which involves dissolving the DSC hard marco and having only the different slicers and the msi_slv_if module as hard macros in the subsystems, the CCI subsystem was chosen to be redesigned adopting this strategy.

#### 6.4.1. CCI Introduction

As shown on **Figure 1**, the i.MX8QM microcontroller has two ARM cores, the A72 and the A53. When in a unique system, different cores keep copies of data stored on the main memory on their local separate caches, a cache coherence interface (CCI) is needed to make sure each local cache has the same data as the others and as the main memory. This is so important to avoid problems when the main memory changes with respect to the caches data.

The actual design has a die size of 924.8*681.6 µm$^2$. It has 4 hard macro modules (cf. **Figure 17**), which are the digital pll (anamix/da_ip_dig_pll_ln28fdsoi), ssi_ace_slave_ca53/ace_slave, ssi/master_128 and the dsc of type gl. The latter has 7 ssslice, 1 slslice and 1 mslice. 4 types of DSC exist. They all have a hardened msi_slv_if along with a hardonned ssslice on the aon part made with RVT, but they differ by the number and types of slices on the PG part. For each type, a RTL module and a physical IP exist. However, subsystems don't use all the dividers present in their DSCs.

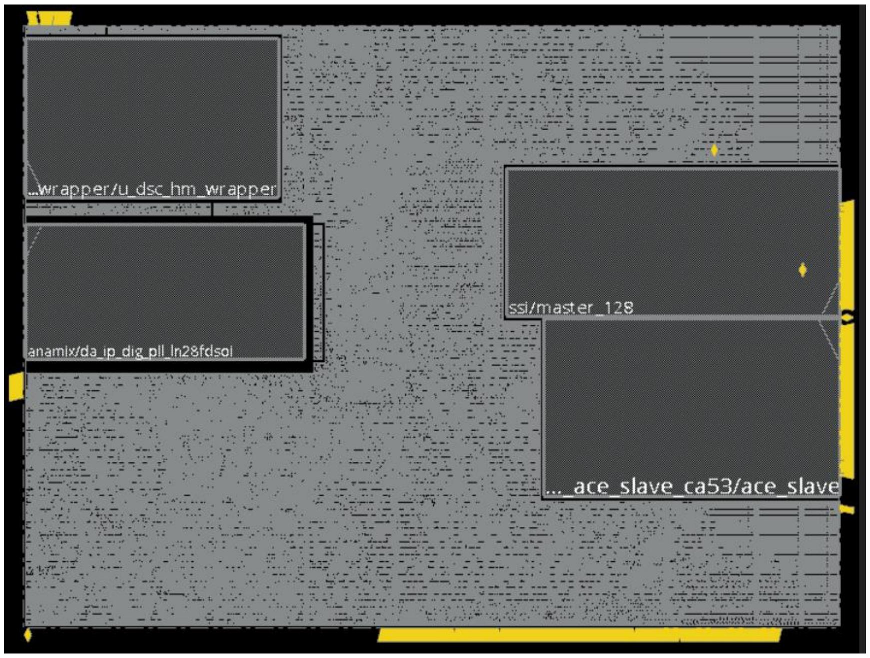

...wrapper/u_dsc_hm_wrapper
anamix/da_ip_dig_pll_ln28fdsoi
ssi/master_128
..._ace_slave_ca53/ace_slave

*Figure 17: CCI current physical layout*

### 6.4.2. New design

The CCI was first resynthesized using the liberty (.lib) files of the different slices previously designed, those of the msi_slv_if and those of the other hard marcos. The extracted gate-level netlist was then modified manually to map the ssslice implemented in the AON power domain on the ssslice_lr[6]. For the timing constraints, the clocks defined at the output ports of the DSC were redefined at the outputs of the slices to make all the timing paths visible to the tools. The other constraints were kept as they were. Concerning the scan insertion, as the internal design flow of the company handles the insertion for a DSC hard marco, the scan was inserted in the CCI and disabled only in the DSC. This design choice was taken to avoid changing the flow which would have taken a lot of time. At the same time, inserting the scan everywhere else allows to have a fair comparison on the area and the easiness it gives to timing closure. However, STA considered only functional mode for timing closure as the scan mode requires a more complete scan insertion.

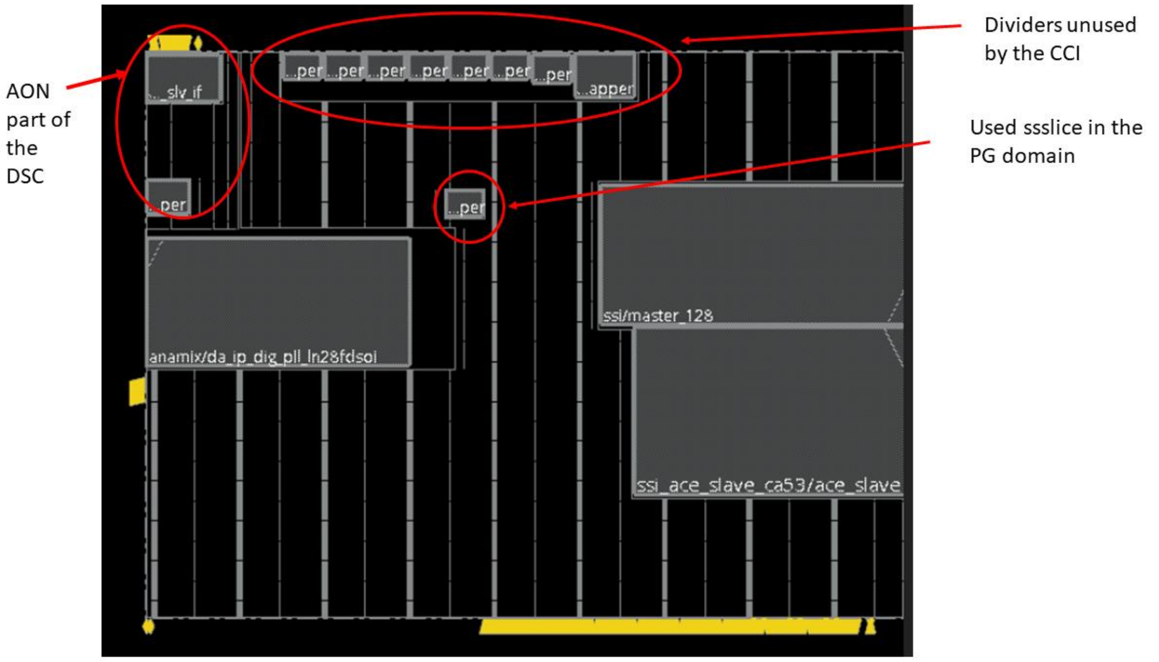

AON part of the DSC
Dividers unused by the CCI
Used ssslice in the PG domain
..._slv_if
..per
...apper
ssi/master_128
anamix/da_ip_dig_pll_ln28fdsoi
ssi_ace_slave_ca53/ace_slave

*Figure 18: CCI new floorplan*

For the physical design, the new design retains the same die size to have a fair comparison with the current physical implementation. All the hard Marcos (except the

[6] Ssslice made with RVT cells to be used on the AON part of the dsc.

DSC) are placed exactly in the same locations. The other placed hard macros are the DSC's ones. The msi_slv_if and the ssslice_lr are placed on the left size of the die to be close to the pll, to have a simple shape for the AON power domain. The msi_slv_if is placed close to the input pins to respect a design requirement of this hard Marco. For the slices used on the PG part, as the subsystems don't use all of them, an analysis was done to determine which ones are used by the CCI. It involved running a STA on the current design and extracting by a script all the flip-flops clocked each slice. The results showed that only one ssslice is clocking sequential elements in the CCI, which means it is the only one used. Based on this conclusion, this ssslice was advantaged and placed close to the output of the pll to reduce the clock latency. The unused slices were placed on the top of the design in order not to block neither the cells placement nor the routing.

The CTS step was a critical one for the physical design and it required many loops to be done. The tool was constrained to:

- Stop scan clocks on the input of the muxes of selection between functional and scan clocks. This avoid over constraining the design and having long buffering chains.
- Ignore the input clock pins of the unused slices in the balancing step while creating the clock tree, to avoid a long run time for unused slices, but also to avoid increasing the size of the tree due to these slices. This strategy required adding false paths to flip flops inside them. Otherwise false hold violations are reported in STA. They are false as these slices are not used.
- Take in account the clock tree already present in the ssi/master_128 as a lot of paths with setup violations were coming from inside it. This involved defining an insertion delay on its input clock pin. The used value was 0.5 ns. It was determined by the different design iterations and was guided by the STA.

The tool showed an abnormal behavior. It routed CTS nets over the msi_slv_if and the pll hard-macros creating DRC violations, but also creating high DRV [7]timing violations. To avoid having these violations the following design steps which leads to high run time and disorients the tool from the real violations to be dealt with, a manual design step was added after the CTS step. It involved deleting all these nets and the re-routing them using the command "ecoRoute". However, this increases the timing the DRV violations. So, to deal with them, a script was developed to extract these nets names in an output file ("cts_deleted.nets") and then it used the command

[7] DRV timing violations is the term that gathers max-transition and max capacitance timing violations.

"optDesign -expandedViews -drv -selectedNets cts_deleted.nets". This command adds an optimization step to these nets concerning DRV violations.

### 6.4.3. Performances Assessment

After timing closure was reached and all timing violations were cleaned, an assessment was run on the new design and the old design to compare them both. This assessment was run using Tempus after running an STA on both designs. The chosen criteria for this comparison were the worst clock latency from the DSC to DFF clock pin, the total leakage power, the total number of functionally useful cells (all cells except physical only cells) and their distribution in terms of vt (P0, P4, …) and the total area occupied by these cells. Unfortunately, the working frequency of the CCI is fixed and so the new design took the same 500 MHz frequency in account and a potential speed increase cannot be assessed.

The worst-case latency[8] of the redesigned CCI is 1.349 ns while it is 1.516 ns for the actual one. The gain of latency found on the ssslice level is retrieved in the subsystem level. Which is a positive element for timing closure easiness. It also shows that an increase of operating frequency can be opted for.

The results presented on **Table 15** and **Table 16** show that the redesigned CCI slightly improves the results. The leakage power[9] decreased of 6.5% (1.4 mW). And although the number of cells increased, the area occupied by these cells decreased of 2.7% (7744 µm$^2$), which means more cells are used for a smaller area, which is a positive result of the new design. However, these results remain limited.

*Table 15: Analysis Results of actual CCI*

| Cells | | | | | | Leakage Power (mW) | |
|---|---|---|---|---|---|---|---|
| Type | P0 | P4 | P10 | P16 | Total | Sequential | 3.8 |
| Number | 4442 | 27583 | 12031 | 184789 | 229014 | Combinational | 17.7 |
| Percentage | 2% | 12% | 5% | 81% | 100% | | |
| Area (µm^2) | 8665 | 37313 | 23236 | 216902 | 286116 | Total | 21.5 |
| Percentage | 3% | 13% | 8% | 76% | 100% | | |

[8] Latency is report is reported for PVT corner (Worst case: 0.9V, -40°C, cmax): Worst corner for timing.
[9] Leakage power is reported for PVT corner (Best case: 1.1V, 125°C, rcmax): Worst corner for power.

*Table 16: Analysis results of redesigned CCI*

| Cells | | | | | | Leakage Power (mW) | |
|---|---|---|---|---|---|---|---|
| Type | P0 | P4 | P10 | P16 | Total | Sequential | 3.3 |
| Number | 2696 | 38747 | 10294 | 191666 | 232951 | Combinational | 16.8 |
| Percentage | 1% | 17% | 4% | 82% | 100% | | |
| Area (µm^2) | 6410 | 38747 | 19356 | 213859 | 278372 | Total | 20.1 |
| Percentage | 2% | 14% | 7% | 77% | 100% | | |

# 7. Conclusion

## 7.1. Technical Conclusion

In conclusion, the decrease of hardening inside the DSC appears to be more beneficial than the actual design strategy of the DSC. The flattening of the ssslice resulted in 30% decrease of latency, 50% decrease of the leakage power and 15% decrease of area. These elements are extremely positive. However, unfortunately, the results on the subsystem level in the case of the CCI were not as relevant as on the ssslice level. The leakage power decreased of only 3.3% and the useful area in the subsystem decreased of only 2.7%. However, the limited results are because the redesigning of the CCI was planned to be as a proof of concept and some elements were kept as they were in the actual design to be able to compare, eventually all the unused slicers (**see Figure 18**) that occupied area and increases the leakage consumption for nothing.

The design strategy is adopted for the CCI has then a potential that was not exploited in the scope of this internship. This can be done by allowing the removal of the unused slicers. This removal should lead to a direct decrease of useful area by 13442 $\mu m^2$ and of the leakage power by 0.74 mW. These values are deduced by summing the results of each unused slicer and they will lead to a total useful area decrease of 7.5% and a total decrease of leakage power by 10%. In addition to the decrease directly coming from removing the slicers, an additional decrease of both criteria will come from the removal of the related buffering cells to these slicers and all the cells used to fix some violations coming from these slicers. In this case, the improvements will start being relevant to the subsystems. It should be added to these elements the easiness that this will give to the time closure which will lead to decreasing the development time.

Therefore, the proposed design strategy for the next projects is to first modify the RTL to allow instantiating only the required slicers in the subsystem level to physically implement only the used slicers. Then from backend perspective, the subsystems should be designed adopting the strategy adopted for the CCI, which involves having only the msi_slv_if, the ssslice of the AON domain and the used slicers of the PG domain as Hard-Macros while the rest of the DSC will be flattened.

As follow-up for the work done during the internship, some points should be reviewed to allow an easy use of this strategy in the following projects.

- The different design flows
- CPF files
- DFT and scan insertion to be adapted for a flat DSC

- For the slicers designed during the internship:
  - Removal of power pins to skip false connectivity errors in Innovus when they are integrated in the subsystems
  - The design should consider unperfect clocks as input of the slicers to model degradation of clock when they are integrated in the subsystems

## 7.2. Personal feedback

This internship has been an experience with many benefits for me.

First, it gave me the chance to discover the world of backend design of which we only see the upper crust during the studies. So, I believe it was a good way to complete my studies path and my last summer internship where I worked more on Frontend and on RTL development. I learned more in depth about synthesis, place and route, STA, physical implementation and physical checks. And I found a lot of interests into this domain as it gets in touch with different aspects, it requires many skills with different type of expertise, and so it gives the chance to learn new things and self-develop continuously. Furthermore, the project I worked on was well placed in the development plan of the company's projects which was a great motivation for me, besides it allowed me to see its different driving elements.

Second working at NXP gave me an overview about a lot of non-technical aspects that can only be experienced working in big companies. During the internship, I was exposed the internal organization of the company and the cooperation between the different teams. I was also lucky to be present during the tape-out phase of a project and the initiation of its following project. So, I had the chance to see the change of work load on the different team members and how the management does to well drive these critical phases. In addition, I was exposed to the fact that the technical feasibility is not always the main driving element to the projects, but there are also the marketing and financial requirements, but there are also the client requirements. The site of Mougins is multi-national site, which allowed me to deal with people from different cultures and ways of working. The site is also co-work with other international sites (China, United States, Brazil, …), therefore it was also great to see how they divide and organize the work together on the same projects.

## 8. References

[1] "About NXP," 09 04 2018. [Online]. Available: https://www.nxp.com/about/about-nxp/about-nxp:ABOUT-NXP. [Accessed 09 04 2018].

[2] L. Leconte, "i.MX8QM top level architecture," Internal Design specification report, Mougins, 2016.

[3] L. Leconte, "NXP i.MX8 power architecture," Internal Design specification report, Mougins, Feb 2018.

[4] M. Elsasser, "DSC_QM_0.5.7," Internal Design specification report, 2016.

[5] M. J. Amatangelo, "Static & Statistical Timing Analysis," The University of Texas at Austin, Austin, 2015.

[6] Samsung, "FDSOI design methodology," Process documentation, 2014.

[7] S. Mukundan, "Physical Only Cells; Well Taps & Decap Cells," 22 03 2018. [Online]. Available: http://vlsi.pro/physical-only-cells-welltap/. [Accessed 11 05 2018].

[8] S. Mukundan, "Physical Only Cells: Filler Cells," 23 03 2018. [Online]. Available: http://vlsi.pro/physical-only-cells-filler-cells/. [Accessed 2018 05 11].

[9] "VLSI System Design," [Online]. Available: http://vlsisystemdesign.com/eco.php. [Accessed 11 05 2018].

## Main Markets

## Internship Placement

RTL codes

↓

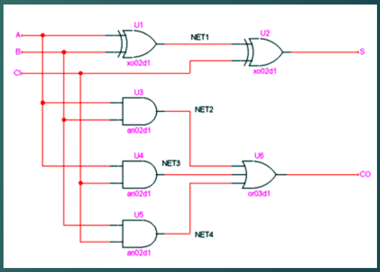
Gate level Netlist

↓

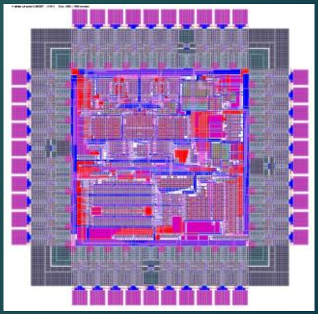
Layout

Digital Backend Design Team

Synthesis tool



Place & Route



By Omar SAKR

## Subject

- Analyzing the current design of the DSC (The clock generator block for i.MX8 microcontrollers)
- Redesigning the DSC exploiting different design strategies
- Main Goal : Decrease the clock latency

Leads to

Less leakage power, smaller useful area, less timing violations, so shorter development time

## Results

- On Block level :
  - 30 % less latency
  - 15 % area gain
  - 50 % less leakage power
- On Subsystem level :
  - 30 % less latency
  - 7.5 % area gain
  - 10 % less leakage power