IMPLEMENTATION FEASIBILITY OF AN INTEGRATED LPDDR4 PHY BLOCK

A Master's Thesis

Submitted to the Faculty of the

Escola Tècnica d'Enginyeria de Telecomunicació de Barcelona

Universitat Politècnica de Catalunya

by

Pol Codina Vilanova

In partial fulfilment

of the requirements for the degree of

MASTER IN ELECTRONIC ENGINEERING

Advisor: Francesc de Borja Moll Echeto

Barcelona, June 2022

Title of the thesis: Implementation feasibility of an integrated LPDDR4 PHY block

Author: Pol Codina Vilanova

Advisor: Francesc de Borja Moll Echeto

Abstract

One of the bottlenecks in the performance of academic RISC-V ASIC processors is high-speed memory access. The use of high speed DDR RAM chips on the board requires the integration in the ASIC of a very complex physical interface block (PHY) that encompasses analog and digital parts. This PHY block is thus technology-specific and very expensive to acquire. Recently, Wavious Ltd. published an open-source description of an LPDDR4x and LPDDR5 with an Apache license containing the digital part and wrappers for the analog parts. This master's thesis will start from this implementation, and will study the feasibility and cost of implementation of this IP for the Barcelona Supercomputing Center RISC-V processor initiative.

Dedicated to all my family and specially to my parents and my grandparents who raised me and educated to be the man I am today. Also, to my friends from high school and college who helped me over the years.

Acknowledgements

I wish to express my sincere thanks to all the team members of the BSC, in particular to the members of the computer sciences department for all their knowledge and previous work.

I also thank to my Project Supervisor, Francesc Moll, for guiding me through my thesis.

And finally, I place on record, my sense of gratitude to one and all who, directly or indirectly, have lent their helping hand in this venture.

Revision history and approval record

Revision	Date	Purpose
0	01/02/2022	Document creation
1	01/07/2022	Document completion

Written by:		Reviewed and approved by:	
Date	01/07/2022	Date	01/07/2022
Name	Pol Codina	Name	Francesc Moll
Position	Project Author	Position	Project Supervisor

Table of contents

List of Figures

List of Tables

1. Introduction

1.1. Context: DRAC project

DRAC is a project to develop a new processor and several open source accelerators [1]. DRAC (Designing RISC-V-based Accelerators for next generation Computers) is a new step in research led by the Barcelona Supercomputing Center (BSC) to manufacture open source chips from Europe. The project has the collaboration of the Universitat Politecnica de Catalunya (UPC), the Universitat de Barcelona (UB), the Universitat Autònoma de Barcelona (UAB) and the Universitat Rovira i Virgili (URV).

Both the processor and accelerators will be based on the RISC-V technology, which is the architecture with ISA (nomenclature that refers to the Instruction Set Architecture) open source chosen by the BSC for Lagarto and for the accelerator being developed in the EPI (European Processor Initiative) project.

1.2. BSC Lagarto ASIC

The first ASIC to be manufactured as part of the DRAC project was preDRAC Figure 1, containing a Lagarto processor.

Lagarto, built with TSMC's 65-nanometer transistors, is the first open-source instruction set architecture (ISA) processor developed in Spain, coordinated by the Barcelona Supercomputing Center (BSC). The preDRAC chip, was a key step in the center's strategy to become a benchmark in the open-source hardware technologies' field developed in Europe.

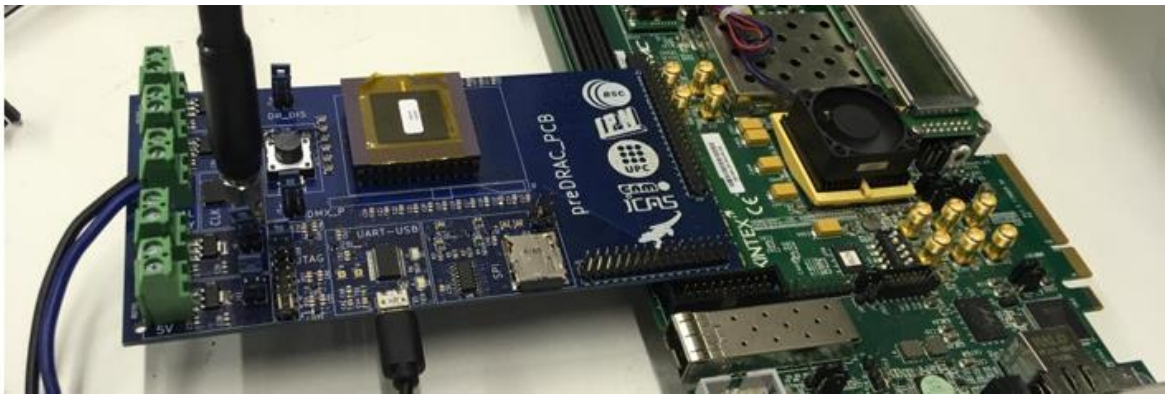


Figure 1 Lagarto chip with the support FPGA

The current implementation of the Lagarto processor needs to be supported by an FPGA that provides access to the DRAM memory as it is shown in Figure 1. The communication is carried out through AXI low speed (32b@50MHz through FMC connector: 200 Mbps), this poses a limitation on the access speed to the DDR3 memory, which has a theoretical performance of 64b@800MHz DDR: 12800 Mbps.

Other communication options are being explored currently that could improve the speed up to 1000 Mbps, but the best solution in terms of performance is an integrated DDR controller to access on-board DDR memory chips.

The physical interface (PHY) between the on-board DDR memory chips and the on-chip memory controller requires a high-speed interface with multiple analog and mixed-signal components. Multiple IP vendors have designs for sale. However it is a very expensive IP to buy (order of 250.000€).

1.3. Wavious open source IP

Wavious [2], a fabless semiconductor company which offers a platform based on mix-and-match "chiplets," has released a RISC-V-powered open-source LPDDR4x/5 PHY – under the permissive Apache 2.0 licence.

The Wavious DDR PHY supports LPDDR4x and LPDDR5 standards and as further explained in section 3.1 is provided in a single 32-bit channel (1x32) configuration (Memory Controller side) consisting of two 16-bit channels (DRAM side). The single 32-bit channel configuration is chosen to mitigate MCU+SRAM area overhead to DRAM interface width.

The Wavious DDR PHY was initially developed in the GF12LPP process (12nm FinFET from Globalfoundries). Although the DDR PHY is highly digital in nature to improve portability to various foundries and process nodes, the DRAM protocol's high transfer rates and varying channel characteristics still require high performance mixed-signal and analog PHY circuits. The Wavious DDR PHY design includes various wrappers to ease standard cell migration and analog models for simulations. In order to port the DDR PHY to other process nodes, designers must perform two tasks. Firstly, designers must incorporate process-specific standard cells into the technology wrappers. Secondly, designers must create analog and mixed-signal schematics/layout which implement high performance circuits.

1.4. Statement of purpose for this thesis

The key points of this thesis are the following:

- Understand LPDDR & DDR PHY
- Understand the structure of the Wavious LPDDR4 PHY.
- Identify necessary analog components and their characteristics.
- Perform a size estimation of the analog components.
- Make a trial physical implementation using black boxes for analog blocks

1.5. Work plan

Task	Duration	Start	End
Research on DDR and LPDDR4	25 days	31/1/22	04/03/22
Signals involved	25 days	31/1/22	04/03/22
Timing and protocols	25 days	31/1/22	04/03/22
MS1: presentation about LPDDR4	0 days	04/03/22	04/03/22
Wavious LPDDR4 IP	30 days	07/03/22	15/4/22
Structure of the digital part	12 days	07/03/22	22/3/22

Structure of the analog part	25 days	12/03/22	15/4/22
MS2: presentation of the digital part	0 days	22/3/22	22/3/22
MS3: List and Specs for analog compoments	0 days	15/4/22	15/4/22
Virtual prototyping with Innovus	40 days	18/4/22	10/06/22
Review hierarchical top-down flow in Innovus	4 days	18/4/22	21/4/22
Study layout of Serdes components	30 days	19/4/22	31/5/22
Synthesis of digital part	10 days	09/05/22	20/5/22
Iterative floorplan	20 days	16/5/22	10/06/22
Estimate timing	15 days	20/5/22 8:00	09/06/22
Top level prototype layout	0 days	10/06/22	10/06/22
Report writing	15,5 days	10/06/22	04/07/22
writing	15 days	10/06/22	01/07/22
MS6: Report	0 days	02/07/22	02/07/22

Table 1 Work plan.

1.6. <u>Deviations from initial plan</u>

The main issue with this project was the lack of documentation from Wavious, this made difficult the understanding of the IP, but also for the moment limits the work that can be done with the analog and mixed signal cells. Also, the implementation process finished with the synthesis of the IP instead of getting to the place and route of the design as initially envisioned.

2. State of the art of the technology used or applied in this thesis

2.1. LPDDR4

2.1.1. DRAM

Dynamic random-access memory (dynamic RAM or DRAM) [3] is a type of random-access semiconductor memory that stores each bit of data in a memory cell, usually consisting of a tiny capacitor and a transistor, both typically based on metal-oxide-semiconductor (MOS) technology. The capacitor can either be charged or discharged; these two states are taken to represent the two values of a bit, conventionally called 0 and 1. The electric charge on the capacitors gradually leaks away; without intervention the data on the capacitor would soon be lost. To prevent this, DRAM requires an external memory refresh circuit which periodically rewrites the data in the capacitors, restoring them to their original charge. This refresh process is the defining characteristic of dynamic random-access memory, in contrast to static random-access memory (SRAM) which does not require data to be refreshed. Unlike flash memory, DRAM is volatile memory (vs. non-volatile memory), since it loses its data quickly when power is removed.

In order to access each storage cell there is a Bit Line Sense Amplifier (BLSA) that will read the small charge in the capacitor and amplify it to a digital signal, after this process the memory will recharge the capacitor. In order to activate a specific cell a decoder assembly will activate the transistor of that cell through each word line. This entire assembly is called a bank shown in Figure 2, RAM chips will feature multiple banks, so each one can be accessed independently of the others, that way it increases overall speed, as one can be read while another one is refreshing.

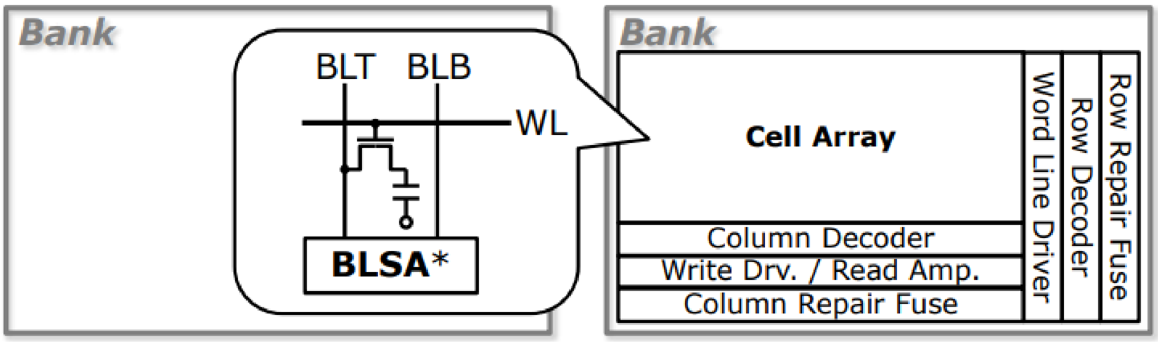


Figure 2 DRAM bank composition [3]

The advantage of DRAM is the density of storage, as a single capacitor and transistor can be integrated in less area than other technologies like SRAM that take 6 transistors [4]. The disadvantage is the refreshing time, although with the use of banks it can be reduced.

Synchronous dynamic RAM (SDRAM) significantly revises the asynchronous memory interface, adding a clock (and a clock enable) line. All other signals are received on the rising edge of the clock. This memory can achieve transfer rates up to 1.3GB/s with a clock speed of 133MHz, this number is dependent on manufacturer optimizations.

Double data rate SDRAM (DDR SDRAM or DDR) [5] was a later development of SDRAM as seen in Figure 3, used in PC memory beginning in 2000. Subsequent versions are numbered sequentially (DDR2, DDR3, etc.). DDR SDRAM uses both the rising and falling edge of the clock and thus internally performs double-width accesses at the clock rate, and uses a double data rate interface to transfer one half on each clock edge, achieving transfer rates up to 3.2GB/s with a clock speed of 200MHz. DDR2 and DDR3 increase transfer speeds up to 6.4GB/s at 400MHz and 14.9GB/s at 933MHz respectively.

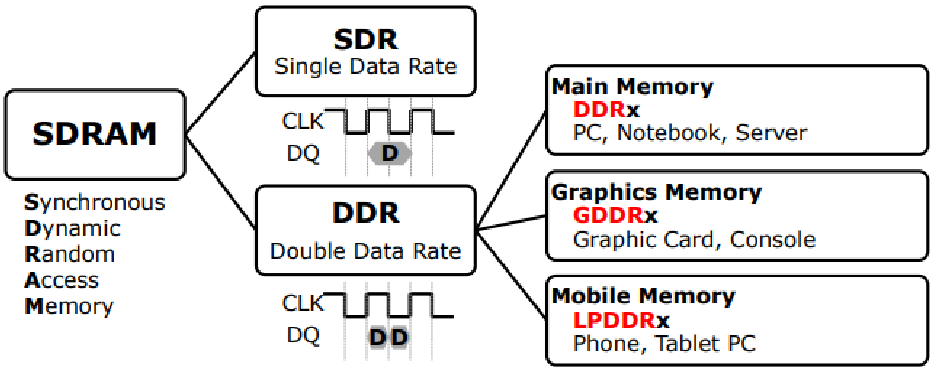
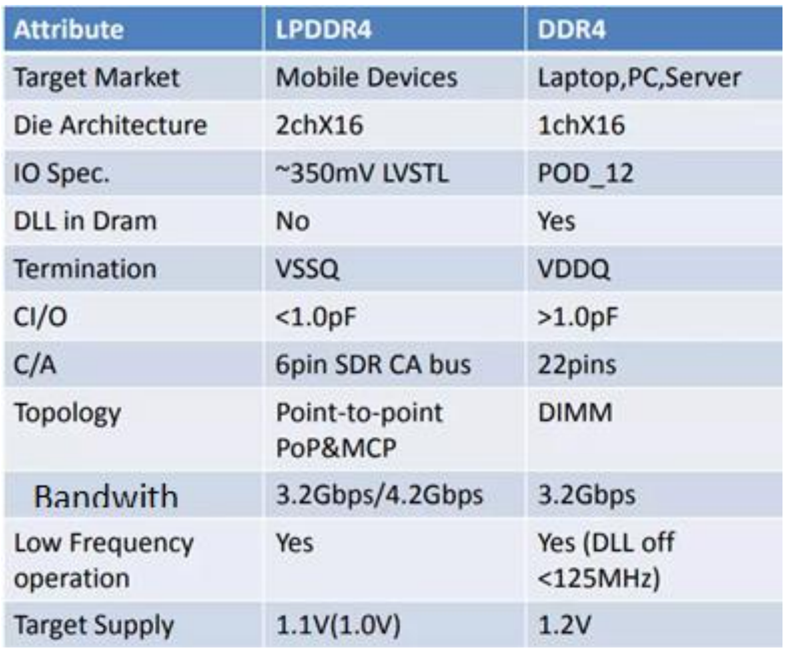


Figure 3 SDRAM development tree [5]

2.1.2. LPDDR4

LPDDR4-SDRAM [6] is a high-speed synchronous DRAM device intended for low power and portable devices, it is designed to reduce power consumption and simplify on-board connectivity. Internally each memory chip is configured as 2 channels, each channel provides an interface that can be accessed independently in order to increase the access speed. Each chip is specified for a capacity up to 16Gb.

While a DDR4 device has an 18-bit address interface and a dedicated 4-bit command interface, the LPDDR4 devices combine on a single 6-bit interface the commands for the memory shown in Table 3 and the address for the data (row and column) as compared in Table 2, this is achieved because the LPDDR4 uses a 2 or 4 clock architecture on the Command/Address (CA) bus, this means that each command or address is divided in 2 or 4 chunks that are transmitted on 2 or 4 clock periods. These devices use a double data rate architecture on the data (DQ) pins to achieve high speed operation.



Attribute	LPDDR4	DDR4
Target Market	Mobile Devices	Laptop,PC,Server
Die Architecture	2chX16	1chX16
IO Spec.	~350mV LVSTL	POD_12
DLL in Dram	No	Yes
Termination	VSSQ	VDDQ
CI/O	<1.0pF	>1.0pF
C/A	6pin SDR CA bus	22pins
Topology	Point-to-point PoP&MCP	DIMM
Bandwith	3.2Gbps/4.2Gbps	3.2Gbps
Low Frequency operation	Yes	Yes (DLL off <125MHz)
Target Supply	1.1V(1.0V)	1.2V

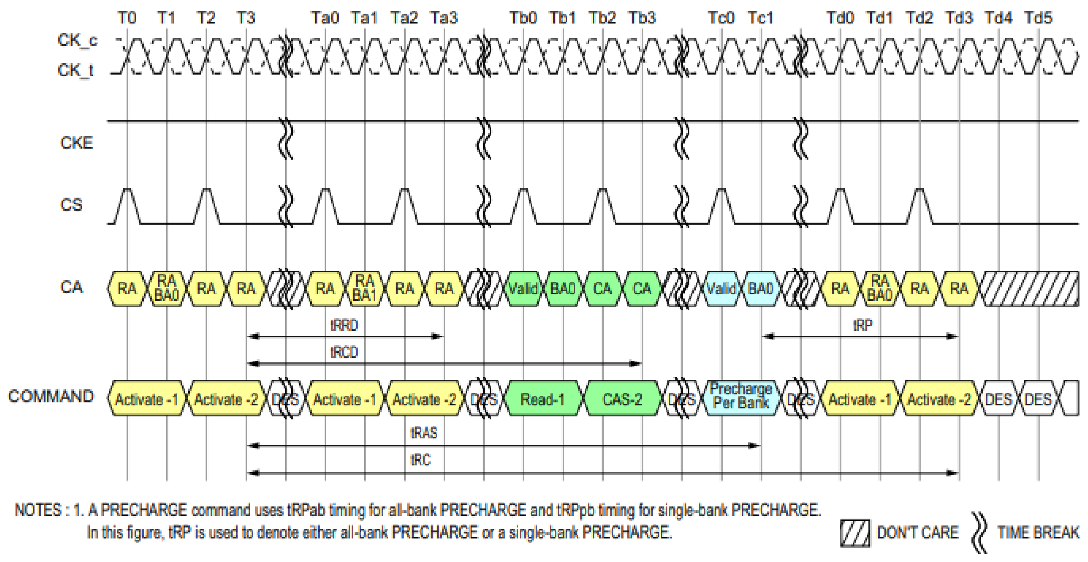
Table 2 LPDDR4 vs DDR4

2.1.3. LPDDR4 operation

Prior to normal operation, the LPDDR4 SDRAM must be initialized [7]. The following provides detailed information covering device initialization, register definition, command description and device operation as shown in Figure 4.

Activate Command

The ACTIVATE command is composed of two consecutive commands, Activate-1 command and Activate-2. The bank addresses BA0, BA1 and BA2 (3-bit address gives access to 8 banks) are used to select the desired bank. Row addresses are used to determine which row to activate in the selected bank. The ACTIVATE command must be applied before any READ or WRITE operation can be executed. After a bank has been activated, it must be precharged before another ACTIVATE command can be applied to the same bank.

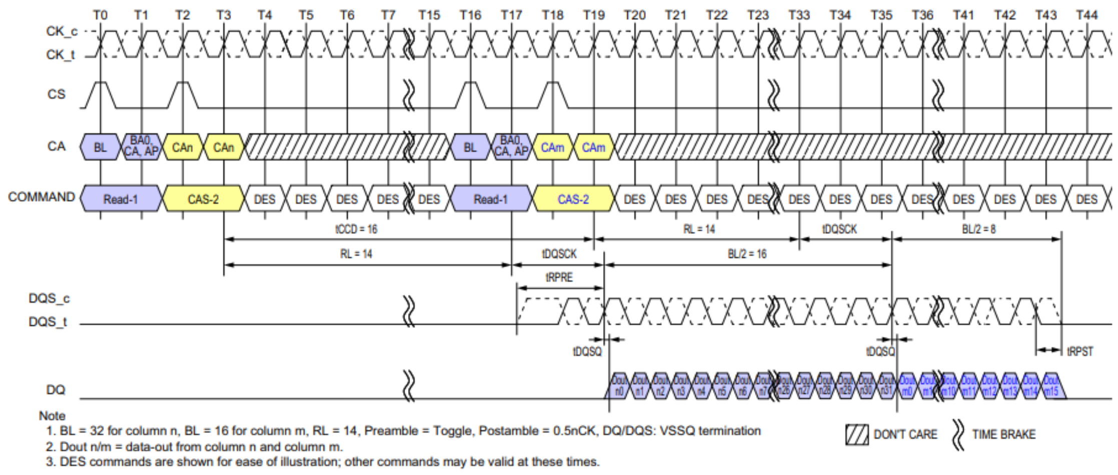


NOTES : 1. A PRECHARGE command uses tRPab timing for all-bank PRECHARGE and tRPpb timing for single-bank PRECHARGE. In this figure, tRP is used to denote either all-bank PRECHARGE or a single-bank PRECHARGE.

Figure 4 Activate operation [7]

Burst Read Operation

After a bank has been activated, a read or write command can be executed. This is accomplished by asserting CKE asynchronously, with CS and CA[5:0] set to the proper state as shown in Figure 5. The LPDDR4-SDRAM provides a fast column access operation. A single Read or Write command will initiate a burst read or write operation, where data is transferred to/from the DRAM on successive clock cycles.



Note
1. BL = 32 for column n, BL = 16 for column m, RL = 14, Preamble = Toggle, Postamble = 0.5nCK, DQ/DQS: VSSQ termination
2. Dout n/m = data-out from column n and column m.
3. DES commands are shown for ease of illustration; other commands may be valid at these times.

Figure 5 Burst read operation [7]

Finally, a sample of the CA table, that shows a few commands used to operate the memory. Each command is composed of two sections, each one is read during a clock edge, so each operation takes two clock edges to be transmitted as seen previously. They are composed of a combination of a 5-bit encoding to show which command is being sent and the information itself, for example the address itself.

	SDR Command Pins	SDR CA Pins (6)							
SDRAM Command	**CS**	**CA0**	**CA1**	**CA2**	**CA3**	**CA4**	**CA5**	**CK_t edge**	**Notes**
Deselect (DES)	L	X						R1	1,2
Multi Purpose Command (MPC)	H	L	L	L	L	L	OP6	R1	1,2,9
	L	OP0	OP1	OP2	OP3	OP4	OP5	R2	
Precharge (PRE) (Per Bank, All Bank)	H	L	L	L	L	H	AB	R1	1,2,3,4
	L	BA0	BA1	BA2	V	V	V	R2	
Refresh (REF) (Per Bank, All Bank)	H	L	L	L	H	L	AB	R1	1,2,3,4
	L	BA0	BA1	BA2	V	V	V	R2	
Self Refresh Entry (SRE)	H	L	L	L	H	H	V	R1	1,2
	L	V						R2	
Write -1 (WR-1)	H	L	L	H	L	L	BL	R1	1,2,3,6,7,9
	L	BA0	BA1	BA2	V	C9	AP	R2	
Self Refresh Exit (SRX)	H	L	L	H	L	H	V	R1	1,2
	L	V						R2	
Mask Write -1 (MWR-1)	H	L	L	H	H	L	L	R1	1,2,3,5,6,9
	L	BA0	BA1	BA2	V	C9	AP	R2	
RFU	H	L	L	H	H	H	V	R1	1,2
	L	V						R2	
Read -1 (RD-1)	H	L	H	L	L	L	BL	R1	1,2,3,6,7,9
	L	BA0	BA1	BA2	V	C9	AP	R2	
CAS-2 (Write-2, Mask Write -2, Read-2, MRR-2, MPC)	H	L	H	L	L	H	C8	R1	1,8,9
	L	C2	C3	C4	C5	C6	C7	R2	

Table 3 Command truth table [7]

2.1.4. Multiple channel connectivity

LPDDR4 is the first JEDEC specification [8] that specifies two DDR DRAM channels per die, and packages with four LPDDR4 DDR DRAM channels, giving chip architects new options when designing chips that connect to LPDDR4 displayed in Figure 6.

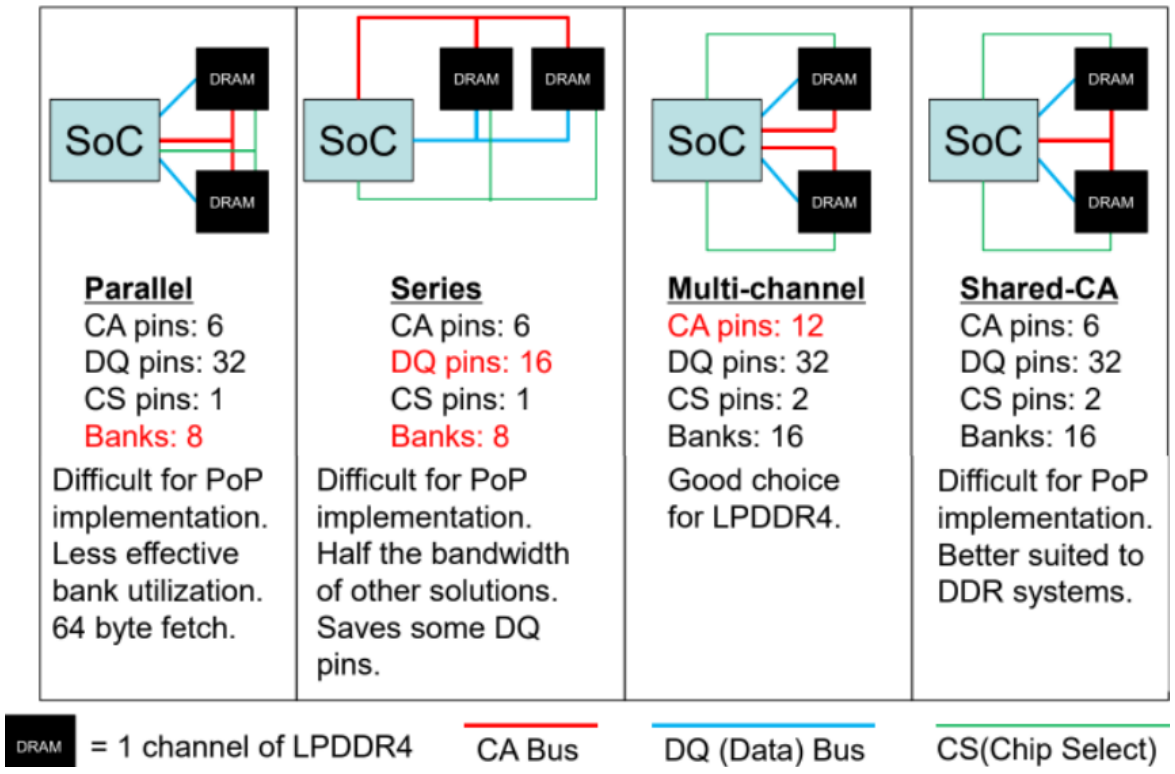


Figure 6 LPDDR4 connectivity options

Parallel connection:

Both DRAM devices receive the same command & address but transmit data over different byte lanes. Both devices are accessed simultaneously.

Difficult for Package on Package (PoP) implementation, it also has less effective bank utilization, has a 64 byte fetch.

Series (Multi-rank) Connection:

Both DRAM devices receive the same command & address and use the same byte lanes. Chip select signals determine which device is being accessed.

Half the bandwidth of other solutions, but saves some DQ pins.

Multi-Channel Connection:

Each DRAM device operates independently of the other, receives a different command & address, transmits over different byte lanes. It is considered the best option for LPDDR4

Multi-channel with Shared-CA (aka Shared-AC):

Both DRAM devices receive the same command & address but only one device is accessed with an active chip select at a time, so each DRAM device operates independently. DRAM devices use different byte lanes, it is fine for DDR4/3, not recommended for LPDDR4.

2.2. Digital implementation

In order to implement high-level designs it is necessary to describe it at a high level of abstraction. For this reason hardware description languages (HDL) like Verilog provide the platform to design complex behaviours. RTL is an acronym for register transfer level, this implies that your HDL code describes how data is transformed as it is passed from register to register. The transforming of the data is performed by the combinational logic that exists between the registers.

Synthesis transforms an RTL code, that is, one using high-level structures like finite state machines, conditional assignments and so on, into a netlist [9]. A netlist is a list of logic gates belonging to a physical library, such that their connection is specified to correctly implement the original RTL description. This netlist is in Verilog language. Therefore, the library information must be provided to the Synthesis tool so that the original description is mapped to the gates corresponding to actual technology. This is referred to as synthesis, the first step of an IC design flow as shown in Figure 7.

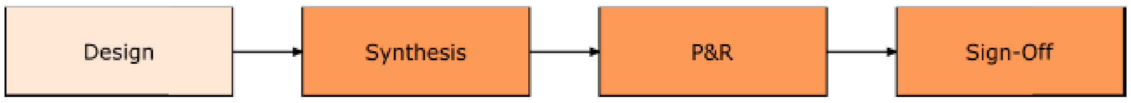


Figure 7 General IC design flow

As mentioned, the synthesis process expects a logical description of a design, but some designs are more analog oriented, this means it is not feasible to use an HDL to design them. Therefore, a synthesis tool is not capable of working with them, these blocks are macros that will be designed using different tools and later included in the design as black-boxes, where the synthesis will connect them to the rest of RTL components. Cadence tool for digital synthesis is called Genus.

In order to avoid typing the same commands time and again, the tool makes extensive use of scripts, written in a standard scripting language, Tool Control Language (TCL). This allows to simplify the procedure by writing several scripts that are modified at convenience without having to type individual commands.

The goal of Genus is to provide the best Quality of Results (QoR) or close to the best with the template flows provided. Still, additional commands and configuration options can be added to improve results. Here, the quality is measured in terms of three aspects: Timing (or performance) given a set of timing constraints written in a special language as described in sections below; Power consumption of the design; and Area of the design.

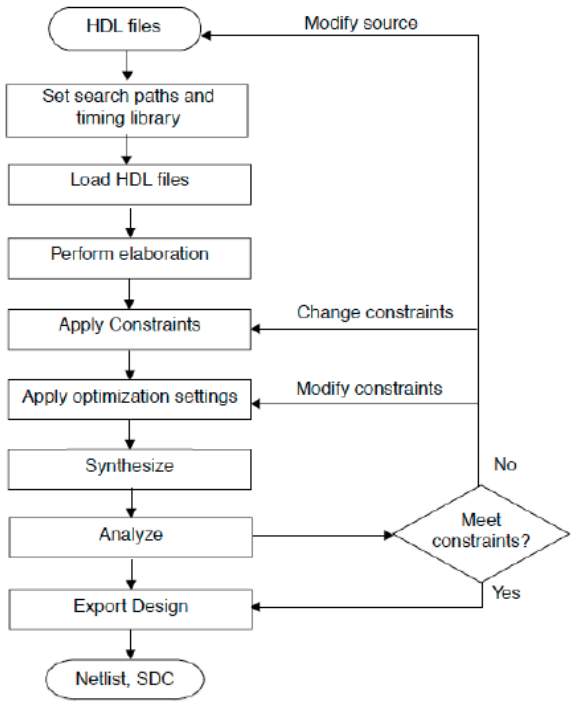


Figure 8 Generic Genus Work Flow

Digital circuits are typically very large in terms of number of transistors and thus a detailed description of their behaviour at electrical level is not feasible. Instead, they are described at logic level, generally describing transactions at the register level, abbreviated as RTL. This description benefits from well-established algorithms that starting from a description in a Hardware Description Language (either VHDL or Verilog) make possible to obtain the optimum number of logic gates implementing the original RTL description. The logic gates used in the synthesis process are predesigned and characterised for a given technology in a so-called Standard Cell library, on which the RTL description is mapped. This process seen in Figure 8 of translating an RTL description into a netlist of logic gates from a library is called synthesis. The gates in the netlist are after that optimally placed on the chip, and the wires routed to connect the gates. This is done automatically with another CAD tool for Place and Route (PnR), Innovus in the case of Cadence.

Design constraints are an essential part of the synthesis process. You apply a set of timing and, optionally, power constraints such that the final design must comply with them. Verifying timing constraints (clock frequency, maximum I/O delays, and so on) are considered the primary goal. No synthesis result is correct until the timing constraints are met. This is measured with a concept called slack. A positive slack means that the timing constraints are met with a given margin (the slack). A negative slack means that the timing constraint is not met, so that netlist must be transformed until the obtained slack is positive. Timing constraints are described in SDC format (a text file).

The final product of the synthesis process is the LEF file. A Library Exchange Format (LEF) file contains library information for a class of designs. Library data includes layer, via, placement site type, and macro cell definitions. This LEF file contains all the data that will later be used to follow with the place and route stage.

2.3. Analog design

However, selected parts of the digital circuit may be designed specifically by hand or with a different design methodology. Typically, analog components and memories belong to this class. They are different from Standard Cells (SC) in which SC represent logic functions that are part of algebraic expressions and so, simplifications and algebraic manipulation with these functions take place when translating from RTL to gates. In addition, all SC in a given library have the same vertical size (thus the name Standard). By contrast, these selected parts, called Macros, have a non-standard size and generally do not represent a simple algebraic expression, but a more complex behaviour. In the RTL they are simply instantiated instead of being described with high level constructs.

The main software used to design analog and mixed-signal systems is Cadence Virtuoso, and Spectre is used to simulate the design. At this stage of the design the aim is to validate the functionality of the design. At this point the design is using an abstract model of the electrical components (Transistors, capacitors…). But, after this stage is completed, the designer needs to implement the design, the layout phase is where the designer will draw the shapes of metal, and different doping areas on the silicon in order to define transistors and the connections between them, also the pins that will connect this design. Finally using abstraction tools, the layout needs to be converted to another LEF file so the Synthesis tools can use it to include it with the rest of the RTL design.

3. Methodology / project development

The Methodology is included in this chapter and should include all relevant methods that have been used as well as research methods and measurements, software and hardware development, etc.

3.1. Wavious IP

The Wavious IP [10] provides the DDR PHY block, the DDR Physical Interface (PHY) IP provides an interface to external DRAM and is an important component to meet system performance requirements as shown in Figure 9.

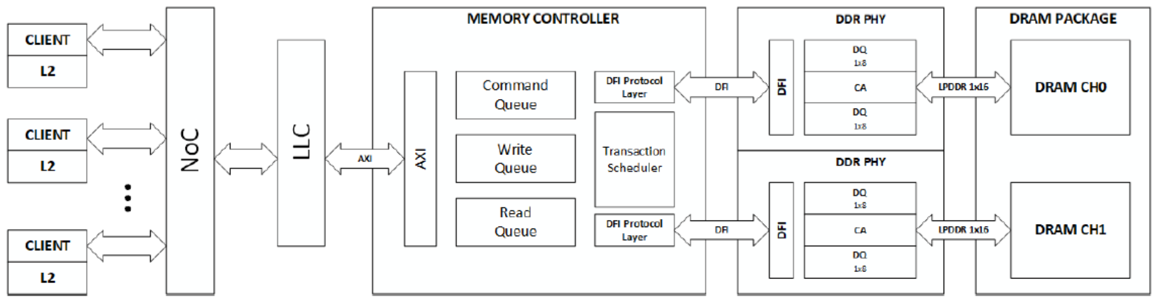


Figure 9 System Block Diagram

The block that we are going to focus on is the DDR PHY interface, inside this PHY we find the following blocks shown in Figure 10.

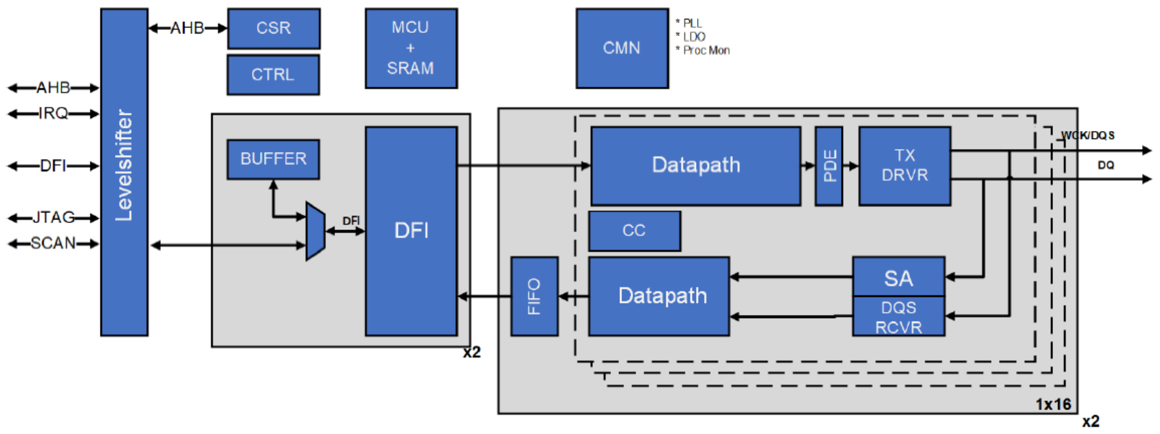


Figure 10 DDR PHY Top Level Block Diagram

The proposed design has an integrated RISC-V MCU and dedicated SRAM for instructions and data storage. Also, it has a common block that contains the PLLs and LDO to provide the clock sources that are needed in this design.

The DDR PHY Interface (DFI) is used in several consumer electronics devices including smart phones. DFI is an interface protocol that defines signals, timing, and programmable parameters required to transfer control information and data to and from the DRAM devices, and between MC (Memory Controller) and PHY.

The DFI interface is not necessary when the MC and PHY are being developed specifically to work together. However, in many situations, the MC and PHY are designed separately – often by different companies. DFI permits companies to develop both MC and PHY IP designs knowing that they will be able to interoperate with the devices developed by other companies.

Additionally, MC devices are primarily digital designs, whereas PHY generally consists of a significant amount of analog logic, therefore the two devices are often developed by different engineers even within the same company. DFI creates a well-defined interface for the two separate design teams.

Finally, it includes the two channels, each one with the TX/RX transceivers.

3.2. Clock structure

Synchronous digital designs are constrained by a clock signal, the period of which establishes the maximum time a signal path can take between two registers. The time that is left between the arrival of the signal to the register and the next edge of the clock signal is the slack, and a positive slack indicates a digital path that is within the time constraints of the system. During the synthesis process Genus will evaluate the timing of every signal path and check the slack, in case it is negative, the designer needs to make changes to the design. For complex designs the clock structure can be composed of multiple clock signals for different domains.

In order to synthesise the Wavious IP we need to define the timing constraints of the different clocks of the design. These constraints are specified in an SDC file. In this file we have to specify all the external clock sources, the generated clocks from PLL, clock gating cells and all the load and IO delays. Since Wavious hasn't provided a clock distribution guide to this day, we will need to use multiple tools to understand the clock structure of the system so it can be specified for the SDC file, in Figure 11 is represented the main clock structure.

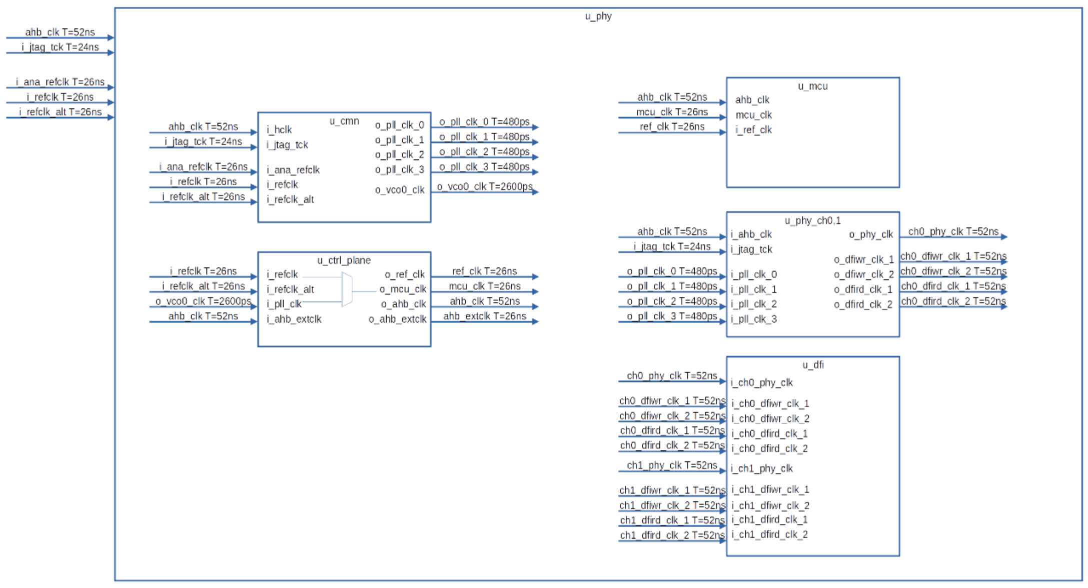


Figure 11 Clock distribution diagram

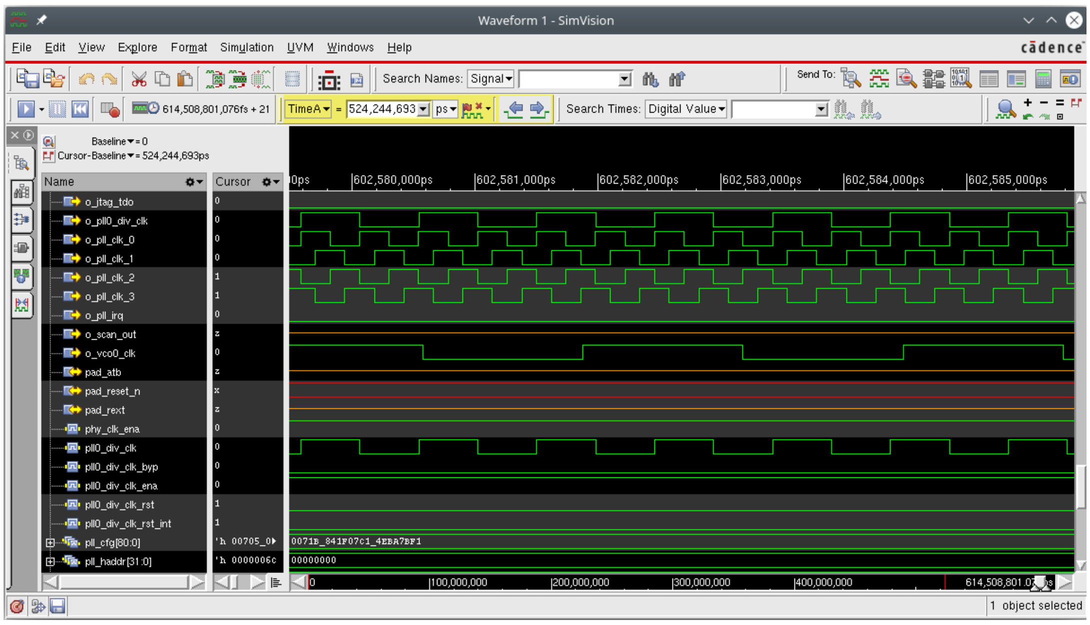
First of all the input clock sources, to find the clock period of these sources we take a look at the simulation testbench that is provided by Wavious with the design, this simulation testbench is intended to validate the design.

```
// Clock Frequencies
localparam AHBCLK_PERIOD      = 10;
localparam REFCLK_FREQ        = 38.4;
localparam REFCLK_PERIOD      = 26;
localparam REFCLK_ALT_PERIOD  = 26;
localparam TCK_PERIOD         = 25;
```

From there we can define the clocks in the SDC file.

```
create_clock -name "Ref_clk" -period 26000.0 -waveform {0.0 13000.0} [get_ports i_refclk]

create_clock -name "Ana_refclk" -period 24000.0 -waveform {0.0 12000.0} [get_ports i_ana_refclk]

create_clock -name "Refclk_alt" -period 26000.0 -waveform {0.0 13000.0} [get_ports i_refclk_alt]

create_clock -name "i_jtag_tck" -period 26000.0 -waveform {0.0 13000.0} [get_ports i_jtag_tck]

create_clock -name "AHB_clk" -period 26000.0 -waveform {0.0 13000.0} [get_ports i_ahb_clk]
```

Some other clock signals are generated by the PLLs, these also have to be specified in the SDC. In order to get the values of these clock periods we can take a look at the simulation results, and measure all the internal clock signals, an example is represented in Figure 12.



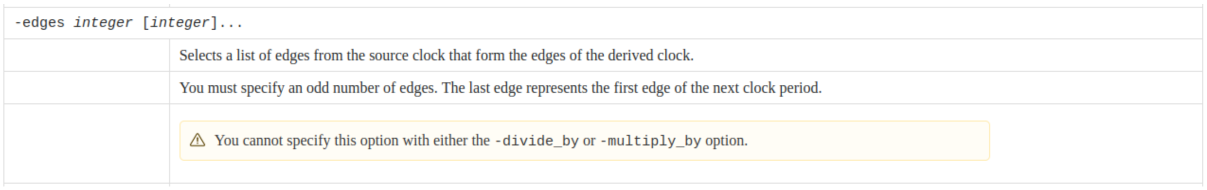
Name	Cursor
o_jtag_tdo	0
o_pll0_div_clk	0
o_pll_clk_0	0
o_pll_clk_1	0
o_pll_clk_2	1
o_pll_clk_3	1
o_pll_irq	0
o_scan_out	z
o_vco0_clk	0
pad_atb	z
pad_reset_n	x
pad_rext	z
phy_clk_ena	0
pll0_div_clk	0
pll0_div_clk_byp	0
pll0_div_clk_ena	0
pll0_div_clk_rst	1
pll0_div_clk_rst_int	1
pll_cfg[80:0]	'h 00705_0
pll_haddr[31:0]	'h 0000006c

Figure 12 SimVision results

The following clock signals of o_pll_clk_0, o_pll_clk_90, o_pll_clk_180, o_pll_clk_270 are generated from the Ref_clk, and have a period transformation and also are phase shifted, these clock signals are used in the analog channels of the PHY interface. To generate these signals we use the following commands.

```
create_generated_clock -add -name "temp_clk" -master_clock [get_clocks Ref_clk] -
source [get_ports i_refclk] -multiply_by 325 [get_pins u_cmn/temp_clk]
create_generated_clock -add -name "o_pll_clk_0" -master_clock [get_clocks temp_clk]
-source [get_ports u_cmn/temp_clk] -edges {1 7 13} [get_pins u_cmn/o_pll_clk0]
create_generated_clock -add -name "o_pll_clk_1" -master_clock [get_clocks temp_clk]
-source [get_ports u_cmn/temp_clk] -edges {4 10 16} [get_pins u_cmn/o_pll_clk1]
create_generated_clock -add -name "o_pll_clk_2" -master_clock [get_clocks temp_clk]
-source [get_ports u_cmn/temp_clk] -edges {7 13 19} [get_pins u_cmn/o_pll_clk2]
create_generated_clock -add -name "o_pll_clk_3" -master_clock [get_clocks temp_clk]
-source [get_ports u_cmn/temp_clk] -edges {10 16 22} [get_pins u_cmn/o_pll_clk3]
```

Using the "-edges" argument we can create a period conversion and a phase shift by selecting delayed edges of a reference clock signal at a higher frequency, as explained by the Cadence guides shown in Figure 13.


-edges *integer* [*integer*]...

Selects a list of edges from the source clock that form the edges of the derived clock.

You must specify an odd number of edges. The last edge represents the first edge of the next clock period.

⚠ You cannot specify this option with either the `-divide_by` or `-multiply_by` option.

Figure 13 From Cadence Genus Command Reference 21.1

Also in the SDC file it is specified the clock uncertainty and the IO delay, this values are taken from the Lagarto project and are approximations of possible real values.

```
####################################
# setting constraints
####################################
set_clock_uncertainty -hold 1.0 [get_clocks Ref_clk]
set_clock_uncertainty -setup 1.0 [get_clocks Ref_clk]
set_clock_uncertainty -hold 1.0 [get_clocks Ana_refclk]
set_clock_uncertainty -setup 1.0 [get_clocks Ana_refclk]
set_clock_uncertainty -hold 1.0 [get_clocks Refclk_alt]
set_clock_uncertainty -setup 1.0 [get_clocks Refclk_alt]
set_clock_uncertainty -hold 1.0 [get_clocks i_jtag_tck]
set_clock_uncertainty -setup 1.0 [get_clocks i_jtag_tck]
set_clock_uncertainty -hold 1.0 [get_clocks AHB_clk]
set_clock_uncertainty -setup 1.0 [get_clocks AHB_clk]
set_clock_uncertainty -hold 1.0 [get_clocks mcu_clk]
set_clock_uncertainty -setup 1.0 [get_clocks mcu_clk]
set_input_transition 200 [all_inputs]
set_max_transition 100.0 -clock_path [all_clocks]
set_max_transition 100.0 -data_path  [all_clocks]
set_max_transition 100.0 [current_design]
set_max_transition 100.0 [get_ports]
set_load -min  0.1 [all_outputs]
set_load -max 0.5 [all_outputs]
###################################
###IO delays
###################################
set_min_delay  1.0 -from [all_inputs -no_clocks]
set_min_delay  1.0 -to [all_outputs]
set_input_delay -clock [get_clocks Ref_clk] 1.0 [all_inputs -no_clocks]
set_output_delay -clock [get_clocks Ref_clk] 1.0 [all_outputs]
```

Finally, the SDC also contains the specifications for the clock gating cells, [11] Genus needs the specification of the setup and hold times for the CGC represented in Figure 14, [12] also the specification for the maximum delay in the cell. If we load the design we are able to see the structure of a CGC cell, and we are able to identify the components so we get a better idea of which pins have to be specified in the SDC file.

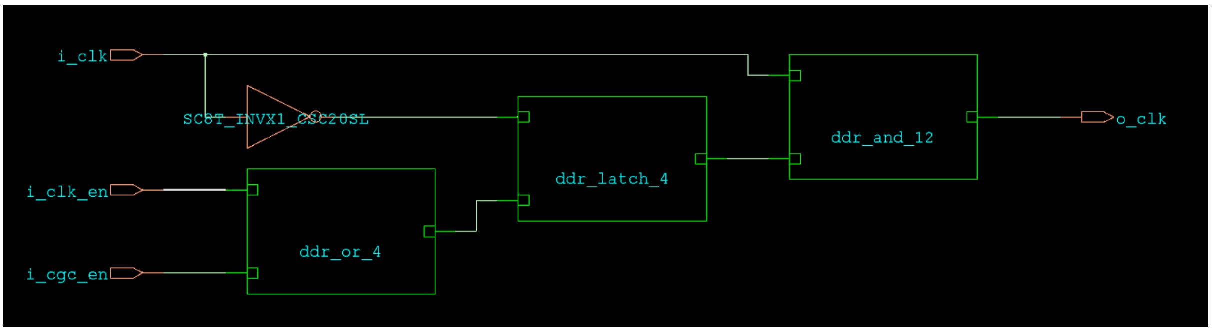


Figure 14 Clock gating cell visualized

```
set_clock_gating_check -high [get_cells
u_phy/u_ahb_ic/u_mcu2ahbic_sync/ASYNC_FIFO.u_ahb_wrd_buf/u_cgc_rl/u_and]
```

```
set_clock_gating_check -setup 1.5 [get_cells
u_phy/u_ahb_ic/u_mcu2ahbic_sync/ASYNC_FIFO.u_ahb_wrd_buf/u_cgc_rl/u_and]

set_clock_gating_check -hold 1.0 [get_cells
u_phy/u_ahb_ic/u_mcu2ahbic_sync/ASYNC_FIFO.u_ahb_wrd_buf/u_cgc_rl/u_and]

set_max_delay 30 -from [get_pins
u_phy/u_ahb_ic/u_mcu2ahbic_sync/ASYNC_FIFO.u_ahb_wrd_buf/u_cgc_rl/u_latch/o_
q_reg[0]/CLK] -to [get_pins
u_phy/u_ahb_ic/u_mcu2ahbic_sync/ASYNC_FIFO.u_ahb_wrd_buf/u_cgc_rl/u_latch/o_
q_reg[0]/Q]
```

3.3. Synthesis process

In order to carry the synthesis, process we will use Tcl scripts developed by the BSC for the Lagarto project as a starting point.

3.3.1. Physical libraries

For the Lagarto project, the BSC uses the 22nm Globalfoundries FDSOI libraries, these contain physical cells for most of the generic components like inverters, latches, logic gates, etc.

Also, for this synthesis process we will use two corners, one typical at 25ºC and 0.8V and one slow at 125ºC and 0.72V. The GF libraries provide models for both corners as well as others not used in this work.

3.3.2. Synthesis issues

Certain unsynthesizable components are present in the code, this is to be expected as there are a lot of Verilog statements that are only meant for simulation, in most cases Genus will ignore these statements and proceed with the synthesis process, still we should understand these statements:

There are *#delays*, that as expected can't be synthetized, as delays are only used for simulations, so Genus will ignore all these.

```
Warning : Ignoring unsynthesizable delay specifier (#<n>) mentioned in verilog file.
These delay numbers are for simulation purpose only. [VLOGPT-35]
```

Various simulation variables used in the design can't be synthetised, some of these variables are only meant for the design stage, but others can be specified when loading the files during the synthesis process.

```
Warning : Unsupported system task or function: assuming value 1'b1. [VLOGPT-31]
        : System function '$test$plusargs' in file '/users/pcodina/wav-lpddr-
hw/rtl/mvp_pll/mvp_pll_sm.v' on line 221, column 41.
    if ($test$plusargs("MVP_FORCE_PLL")) begin
```

Duplicate modules, designed by Wavious, we are not sure as there are no analog specs to differentiate these blocks, so, we let Genus pick one and proceed with the default option.

```
Warning : Duplicate module definition. [VLOGPT-661]
        : Ignoring definition of 'ddr_wcm_inv' module in file '/users/pcodina/wav-lpddr-hw/rtl/wddr/ddr_custom_lib.sv' on line 25, column 18.
        : To modify the behaviour, do : set_attribute hdl_keep_first_module_definition false.
```

Value initialization used for simulation purposes.

```
Warning : Ignoring unsynthesizable construct. [VLOGPT-37]
        : Initial value assignment to reg in file '/users/pcodina/wav-lpddr-hw/rtl/wddr/Wpin_uart_lib.sv' on line 29, column 16.
reg start_bit=1'b0;
```

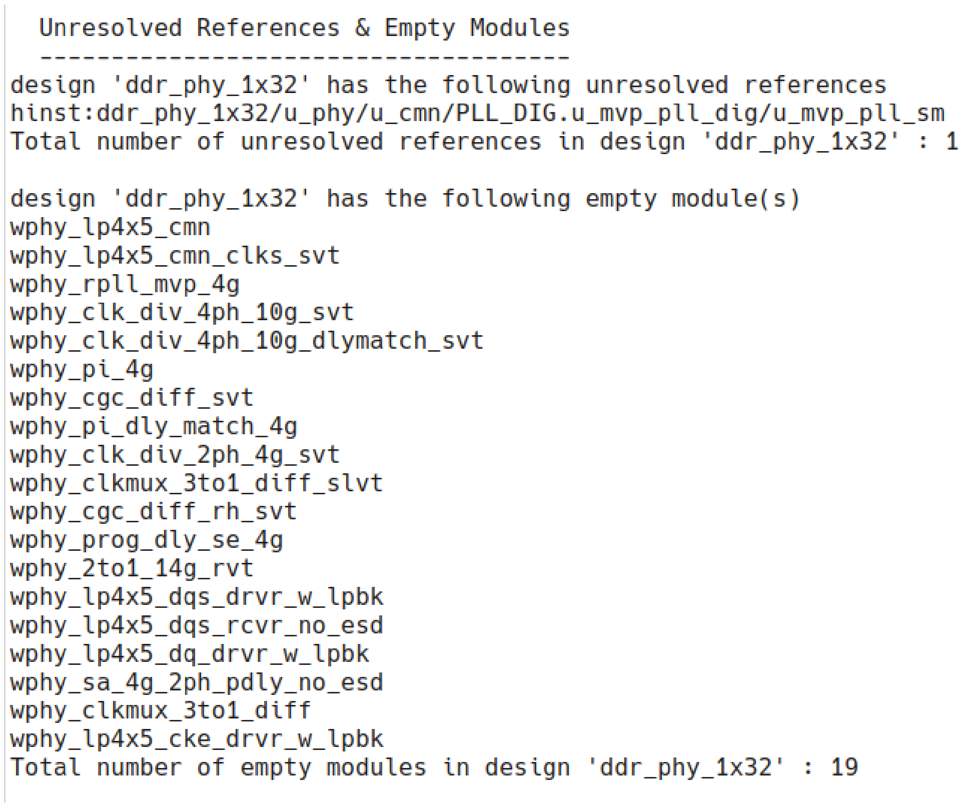
Using $realtobits operations for simulation are not synthesizable, and are only used to display values during simulations.

```
Warning : Unsupported system task or function: assuming value 1'b1. [VLOGPT-31]
        : System function '$realtobits' in file '/users/pcodina/wav-lpddr-hw/rtl/wddr/Wpin_uart_lib.sv' on line 130, column 32.
     #(clk_period/2.0);
```

Files that only contain simulation tools are excluded from the synthesis list (Wpin_uart_lib.sv, mvp_pll_sm.v, ddr_project_vpp.vh).

3.3.3. Empty modules

As expected, once we initiate the synthesis process Genus encounters empty modules, these are design blocks with an analog or mixed-signal design, therefore Genus can’t use the digital libraries GF22FDX, as these only contain digital components. In total Genus found 19 components that can’t be synthesised as seen in Figure 15. In order to complete the synthesis each of these blocks will need to be evaluated individually.



```
  Unresolved References & Empty Modules
  -------------------------------------
design 'ddr_phy_1x32' has the following unresolved references
hinst:ddr_phy_1x32/u_phy/u_cmn/PLL_DIG.u_mvp_pll_dig/u_mvp_pll_sm
Total number of unresolved references in design 'ddr_phy_1x32' : 1

design 'ddr_phy_1x32' has the following empty module(s)
wphy_lp4x5_cmn
wphy_lp4x5_cmn_clks_svt
wphy_rpll_mvp_4g
wphy_clk_div_4ph_10g_svt
wphy_clk_div_4ph_10g_dlymatch_svt
wphy_pi_4g
wphy_cgc_diff_svt
wphy_pi_dly_match_4g
wphy_clk_div_2ph_4g_svt
wphy_clkmux_3to1_diff_slvt
wphy_cgc_diff_rh_svt
wphy_prog_dly_se_4g
wphy_2to1_14g_rvt
wphy_lp4x5_dqs_drvr_w_lpbk
wphy_lp4x5_dqs_rcvr_no_esd
wphy_lp4x5_dq_drvr_w_lpbk
wphy_sa_4g_2ph_pdly_no_esd
wphy_clkmux_3to1_diff
wphy_lp4x5_cke_drvr_w_lpbk
Total number of empty modules in design 'ddr_phy_1x32' : 19
```

Figure 15 Unresolved references found by Genus

3.4. <u>Full Custom blocks: analog and mixed-signal</u>

As mentioned previously, the Wavious IP contains placeholders for the analog and mixed signal circuits that need to be designed to work in our technology. Due to the number of circuits that need to be designed and the difficulty of it, for the purpose of this thesis we will only evaluate each block and attempt to estimate the area, that way we will design a set of black boxes so when we finally synthesise the design, the area calculation is realistic.

3.4.1. wphy_lp4x5_cmn_clks_svt.sv **- Common clock driver**

This module is the output driver for the clocks generated in the common block, therefore the design will require an emphasis on current delivery and precise timing, so it needs an analog design. If we look at the logical description, we are able to get an idea of the functionality of the design, this will give a general idea of the required components of the system, we find the following blocks:

1 latch

18 inverter

2 flip-flop

3 Clock Gating Cells (CGC)

1 wphy_lp4x5_cmn_clks_svt_wphy_clk_div2_4g_core_svt (detailed in 3.4.1.1)

2 wphy_lp4x5_cmn_clks_svt_wphy_gfcm_svt (detailed in 3.4.1.2)

3.4.1.1. wphy_lp4x5_cmn_clks_svt_wphy_clk_div2_4g_core_svt

This module contains the core logic of the clock driver, it has the following components:

1 NOR

12 inverter

8 inverter with enable

2 latch

3.4.1.2. wphy_lp4x5_cmn_clks_svt_wphy_gfcm_svt

This module has the same function as the wphy_gfcm_lvt.sv - Glitch free clock mux. It contains:

20 inverters

2 NOR

6 latch

3 NAND

In total if we add all the areas of the components together, we could estimate an area of 50µm^2.

3.4.2. wphy_rpll_mvp_4g.sv **- Multi-VCO PLL**

A PLL is a device that using a reference clock signal will create a new one with a different frequency while keeping the phases synchronised, this requires precise timing and a design focused on current delivery, therefore these components are designed with an analog workflow.

According to reference [13] that designs a PLL using a 22nm technology we could get an approximation of the area, the PLL area is 0.015mm2 (143µm x 105µm) as seen in the micrograph of Figure 16. This design is in a different technology, but as mentioned before this is an initial estimation.

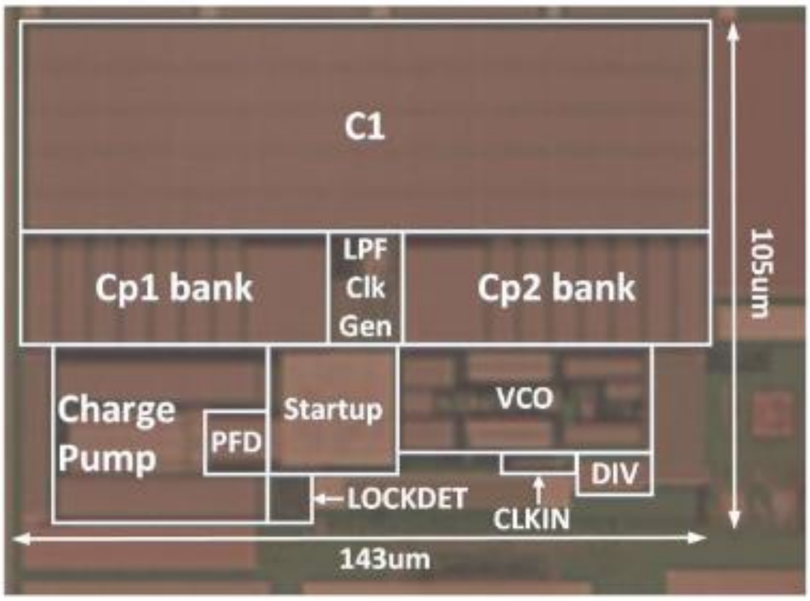


Figure 16 22nm PLL [13]

3.4.3. wphy_clk_div_4ph_10g_dlymatch_svt.sv - **4 phase clock divider delay match**

A 4-phase clock divider will create new clock signals with a phase shift from a reference clock signal. Although the design of the block seems to be digital due to the high speed and the current delivery requirements of a clock driving component the design needs to be done using a full custom analog workflow. To estimate the area of this component we will add the area of all the components that are used in the Verilog.

21 inverter with enable
4 latch
28 inverter
4 X2 inverter

In total if we add all the areas of the components, the approximate area of the block is 20µm^2.

3.4.4. wphy_clk_div_4ph_10g_svt.sv **- 4 phase clock divider**

This component will have a similar functionality as the one with delay match. To estimate the area of this component we will add the area of all the components that are used in the Verilog.

1 NOR
12 inverter
6 inverter with enable
2 latch

In total if we add all the areas of the components, the approximate area of the block is 10μm^2.

3.4.5. wphy_pi_4g.sv - **Phase interpolator**

A phase interpolator will generate a clock signal with a controlled phase shift. In order to create this phase shifting the design uses 4 reference clock signals with a 90 degree phase shift that will get interpolated using current control. Therefore this block needs an analog workflow.

The proposed phase interpolator of reference [14] consists of two sets of N-blocks of mixing current sources of phases 0 and 90 degrees, as given in a half-rate receiver and it is shown in Figure 17. In particular, each block has two differential PMOS and NMOS switches at the highest and lowest voltage level, mitigating the effect of body bias and thus making them faster to switch on and off. A main common current source is supplying the current to integrate on the mixing capacitors, and hence the number of the enabled interpolation blocks via control bus, define its integration strength and hence corresponds to a different generated phase.

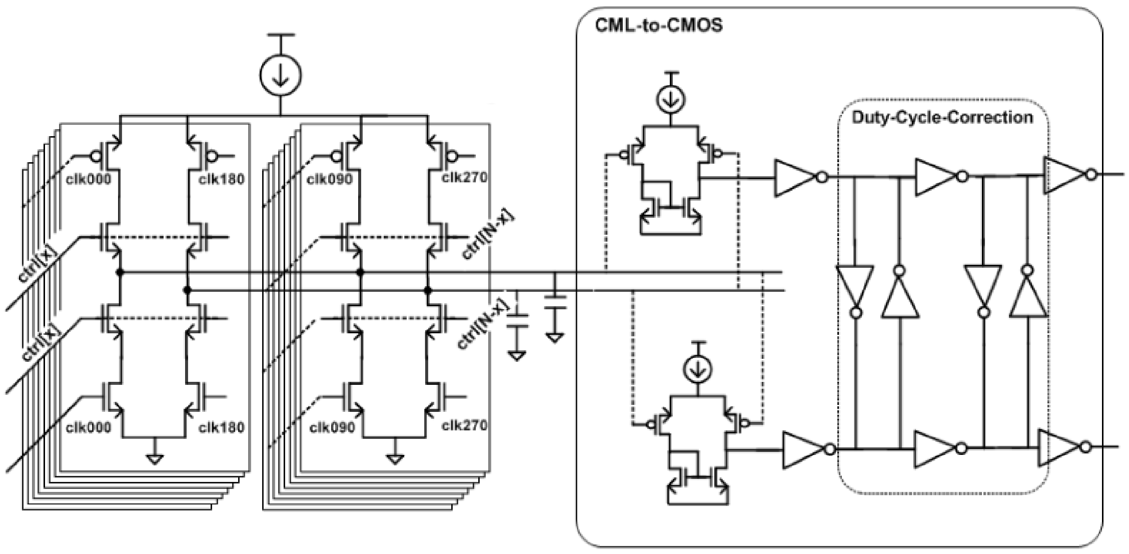


Figure 17 Diagram of proposed topology of two-phase mixing

The input of the CML-to-CMOS block is a simple single stage amplifier, operating in open-loop configuration. It is necessary to convert the low-voltage levels of the mixing stage to a full-scale digital logic signal at the output. Hence, in terms of design requirements, there are no particular tight specification other than enough bandwidth to amplify the capacitor voltage.

Following the amplifying stage, the rectified signal has an asymmetrical duty cycle, and hence a duty cycle correction circuit follows up to obtain a 50% duty cycle clock signal.

In total we are able to count 10 inverters 94 NMOS 34 PMOS and 2 capacitors, looking at our available technology, we estimate an area of 220µm^2, most of it coming from the capacitors, so the shape of the cell could easily be square (15x15µm).

3.4.6. wphy_cgc_diff_lvt.sv - **Differential clock gating cell (rest low)**

A regular clock gating cell Figure 18 has the function of cutting the clock signal from propagating, this way it saves power as it deactivates the dynamic operation of the controlled area of the design.

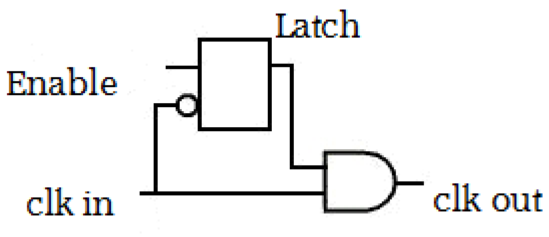


Figure 18 Typical clock gating cell

Due to the high speed requirements of this design, the logical description is more complex than a traditional inverter.

6 inverters

1 latch

2 inverter with enable

In total if we add all the areas of the components together, we could estimate an area of 5µm^2.

Ther are two versions of this component, this one is “rest low”, meaning that when the clock is disabled the output signal is forced to a logical “0”, while the “rest high” version of this component forces a logical “1” at the output.

3.4.7. wphy_pi_dly_match_4g.sv **- Phase interpolator delay match**

Similar design to the regular phase interpolator, so the phase matching would be an extra design requirement, but not add any significant extra area to the design. So, the area would be 220µm^2 (15µmx15µm).

3.4.8. wphy_clk_div_2ph_4g_svt.sv - **2 phase clock divider**

A 2-phase clock divider will create a new clock signal with a phase shift of 180º from a reference clock signal, although the design of the block seems to be digital due to the high speed and the current delivery requirements of a clock driving component the design needs to be done using an analog workflow. To estimate the area of this component we will add the area of all the components that are used in the Verilog.

1 NOR

11 inverter

8 inverter with latch

2 latch

In total if we add all the areas of the components together we could estimate an area of 10μm^2.

3.4.9. wphy_clkmux_3to1_diff.sv - **Differential 3:1 clock mux**

A multiplexer is a component that can switch between to input signals using a control one. Multiplexers can be built in multiple ways, for this application Wavious uses tri-state logic as seen in Figure 19.

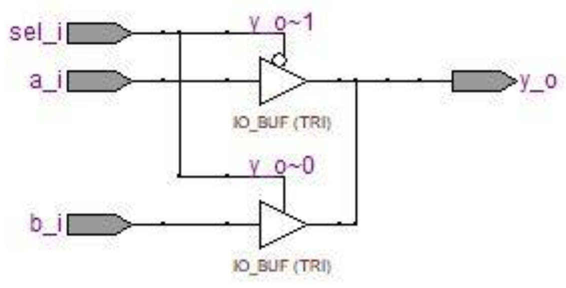


Figure 19 Tri-state multiplexer

27 inverter

20 inverter with enable

4 NAND

Because it uses tri-state logic it requires less additional gates, but because it carries a clock signal and it needs the speed and additional current capabilities it has extra inverters in parallel. In total if we add all the areas of the components together, we could estimate an area of 15μm^2.

3.4.10. wphy_prog_dly_se_4g.sv - **Programmable delay**

The high speed requirements of the PHY interface requires that each communication signal has the same delay, a programmable delay is required to compensate for any difference in the paths of the transmission signals. In order to understand how this block works we can look at the system Verilog description. It can apply a delay to an input signal, and this delay can be configured with two control signals. The range of this delay can go as low as 63ps with a resolution of 1ps at the lowest scale.

```
initial begin
    case(gear)
        0: delay_total = 200+5*i_ctrl;
        1: delay_total = 110+3*i_ctrl;
        2: delay_total = 78+2*i_ctrl;
        3: delay_total = 62+1*i_ctrl;
        default: delay_total = 0.0;
    endcase
```

A first approach would be to have 4 separate delay lines, that are selected with a 2-bit mux controlled by gear signal, multiple examples shown in Figure 20. Each delay line has a fixed time delay, and a controlled delay. These delay lines would be built with inverter chains. But, this design is not suitable, as delay needs a resolution of 1ps, and minimum delay block available in our physical cell libraries is 20ps.

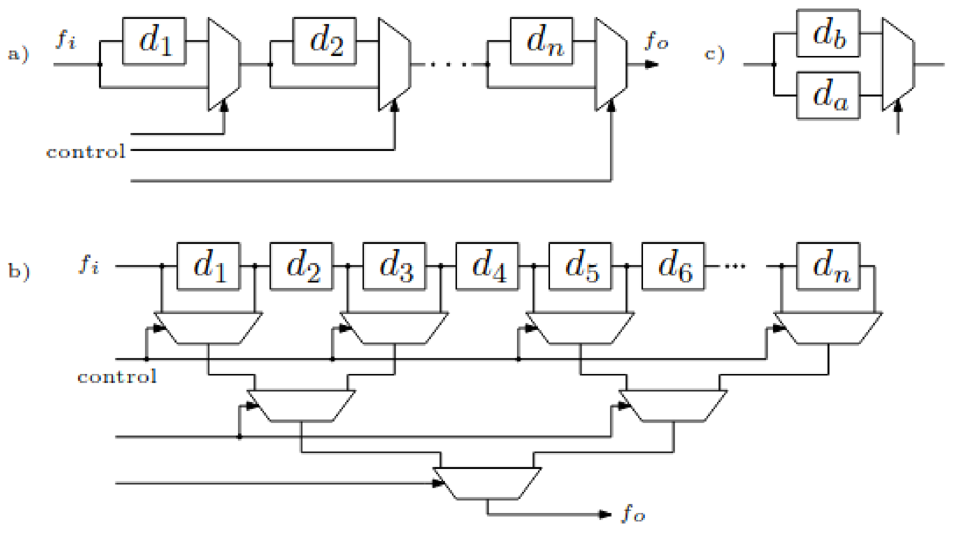


Figure 20 PDC delay line structure: (a) cascaded design (b) tree-like design (c) fine differential delay

So, we propose an analog solution that consists of changing the delay by changing the capacitance load in between inverters, a schematic is represented in Figure 21.

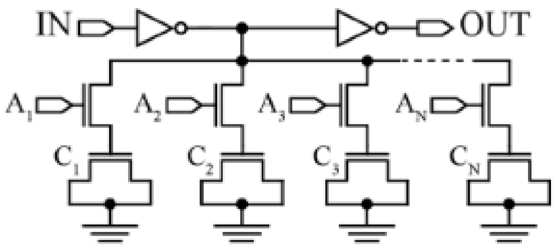


Figure 21 Load based delay line

For each case we have a fixed delay and the programmable delay, the fixed one can be taken from the libraries, and the programmable comes from the proposed solution, because each one has 6bits of resolution, so 6 pairs of MOS and MOS capacitor structures, each capacitor double the size compared to the previous. In total 15μm^2.

3.4.11. wphy_2to1_14g_rvt.sv - **Differential 2:1 serializer**

Typical implementation of a 2:1 serializer uses 3 latches and one multiplexer to feed to signals on a single wire during alternate clock periods as is represented in Figure 22:

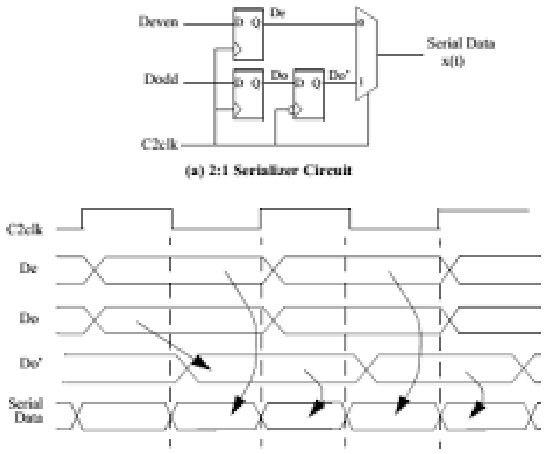


Figure 22 Typical 2:1 serializer

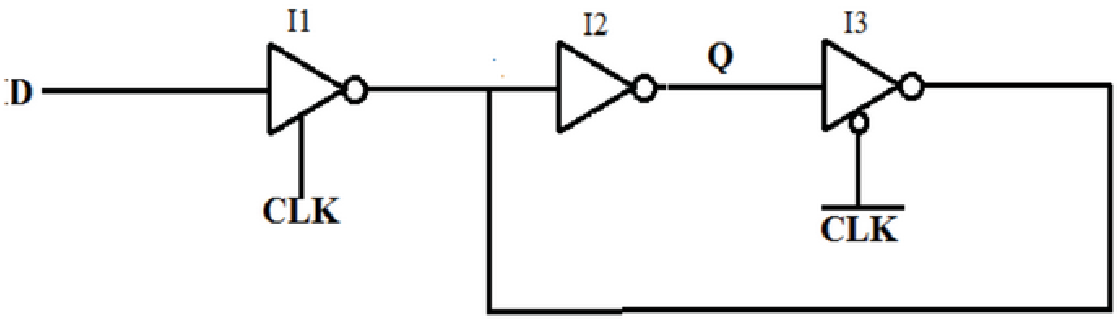


Figure 23 Tri-state latch

But this design uses tri-state latches that can be built, according to the paper [15], using 3 tri-state inverters like in Figure 23. In total if we add all the areas of the components together, we could estimate an area of 7μm^2.

3.4.12. LVSTL pin drivers

Due to the high speed requirements of the LPDDR4 communications the IO pin drivers are required to operate at very specific current levels with a controlled impedance, also the receiver pins will feature custom designed amplifiers.

Due to the complexity of these transceivers it is hard to estimate their area just by looking at the digital description. The proposed solution is to look at another design for a PHY interface for LPDDR4, this design [16] has a total chip area of 12 mm^2 in a 65nm CMOS process, from the micrograph picture in Figure 24 we can estimate the area of the components by measuring and using the Dennard scaling law estimate the area for our 22nm process. Using this method we obtain the following results.

wphy_lp4x5_dq_drvr_w_lpbk:16800μm^2

wphy_lp4x5_dqs_rcvr_no_esd:34000μm^2

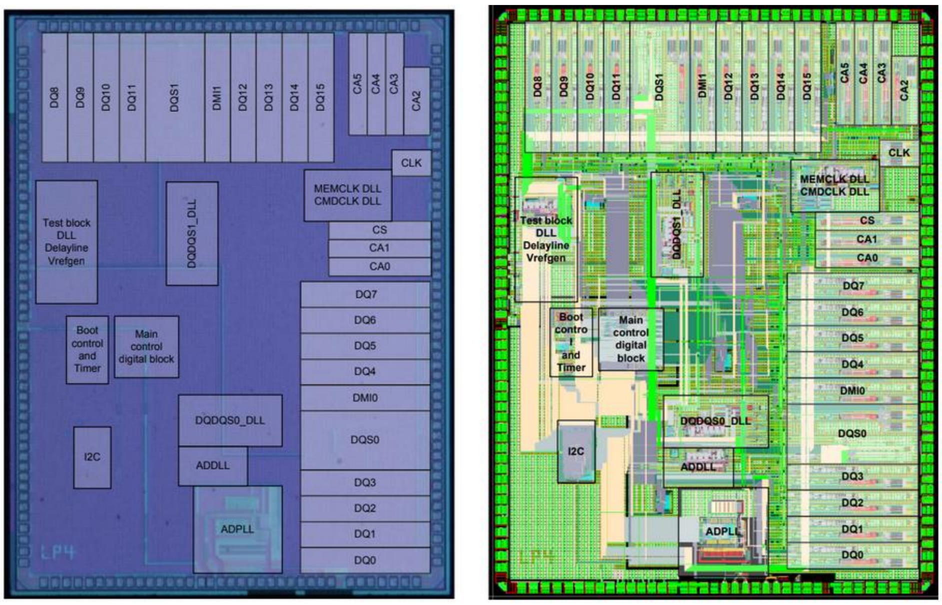


Figure 24 LPDDR4 controller micrograph

3.4.13. wphy_lp4x5_cmn.sv **- Common clock**

Using the same method used to estimate the area for the LVSTL pin drivers (3.4.12) we can obtain an approximation for the common clock block of 55000μm^2.

3.5. <u>Generating placeholders</u>

All the previous full custom blocks need a LEF file description in order to be included in the digital implementation flow so that their area is correctly taken into account. The LEF file description represents the block size with I/O pin locations. For our purpose, since this work scope does not cover the full custom blocks design, we make a simplified design with the minimal physical information to generate the LEF.

First of all using Virtuoso [17] we will create the schematic in Figure 25 a of the design we want to create a placeholder, because we are only creating a black box, the schematic is only the pins of the block shown in Figure 25 b.

In order to create a layout we use the Virtuoso tools to automatically place all the pins on the corresponding metal layers and in the required shape shoen in Figure 25 c, for now it is a square with the dimensions that we have calculated previously.

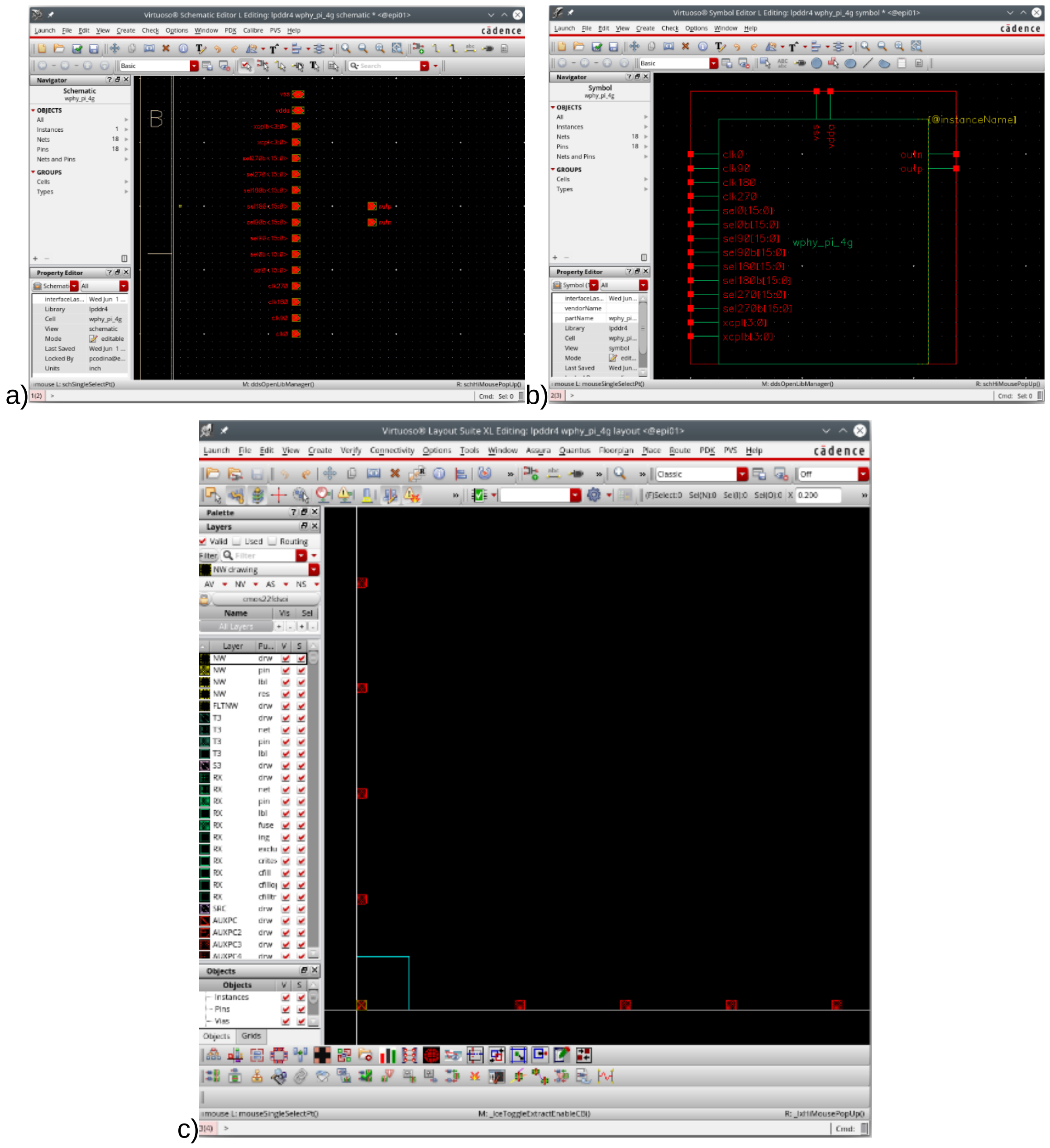
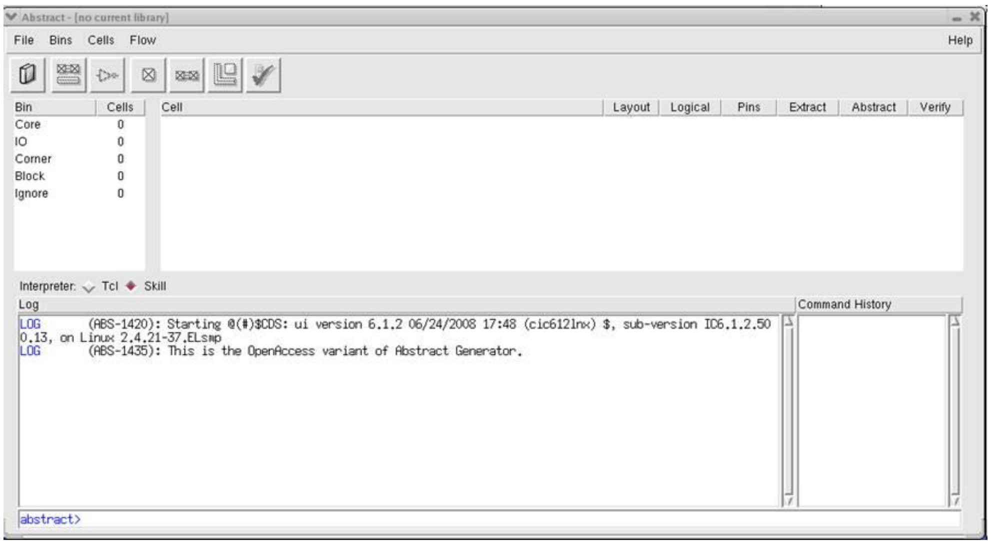

a)
Virtuoso® Schematic Editor L Editing: lpddr4 wphy_pi_4g schematic * <@epi01>
b)
Virtuoso® Symbol Editor L Editing: lpddr4 wphy_pi_4g symbol * <@epi01>
vss
vdda
clk0
clk90
clk180
clk270
sel0[15:0]
sel0b[15:0]
sel90[15:0]
sel90b[15:0]
sel180[15:0]
sel180b[15:0]
sel270[15:0]
sel270b[15:0]
xcpl[3:0]
xcplb[3:0]
outn
outp
wphy_pi_4g
[@instanceName]
c)
Virtuoso® Layout Suite XL Editing: lpddr4 wphy_pi_4g layout <@epi01>
Palette
Layers
Objects
cadence

Figure 25 a)Virtuoso schematics, b) symbol and c) layout

At this point we follow the abstract generator tool shown in Figure 26 from cadence and their guide [18]. With this tool we generate the LEF files that have to be included in the Tcl script so the synthesis process can load the abstracts for the design. Yet another tool should generate LIB files that would contain timing and leakage power consumption of the block, but due to increased complexity of estimating these figures we will leave this step for future work.



Bin	Cells
Core	0
IO	0
Corner	0
Block	0
Ignore	0

Cell	Layout	Logical	Pins	Extract	Abstract	Verify

```
Log
LOG       (ABS-1420): Starting @(#)$CDS: ui version 6.1.2 06/24/2008 17:48 (cic612lnx) $, sub-version IC6.1.2.50
0.13, on Linux 2.4.21-37.ELsmp
LOG       (ABS-1435): This is the OpenAccess variant of Abstract Generator.

abstract>
```

Figure 26 Abstract generator

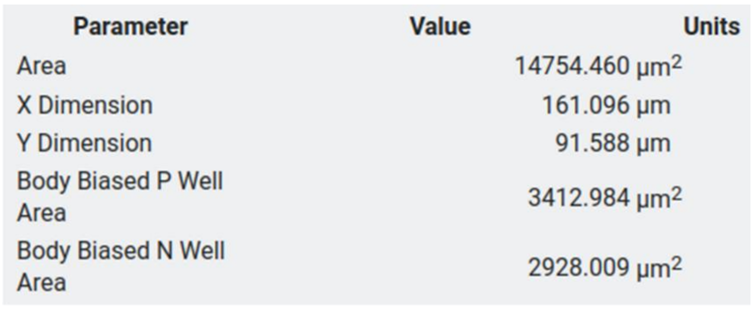
3.6. SRAM

The design includes an SRAM memory to support the integrated RISC-V processor. The specifications of this memory are in the system Verilog.

```
parameter      AWIDTH      = 32,
parameter      DWIDTH      = 32,      // SRAM Width in Bits
parameter      DEPTH      = 2048,     // Maximum SRAM WORDS
parameter      STRB_WIDTH  = (DWIDTH/8),
parameter      SIZE       = 65536,    // TCM Size in KB
parameter      PIPELINE    = 0,
parameter  CDE_FILE_INIT     = "default.hex",
parameter INITIAL_MEM_DELAY = 450
```

The design of an SRAM memory is highly complex, for this reason the foundry provides what is called a memory compiler to generate the physical macros given the memory parameters. So, the process is as simple as feeding the parameters to the scripts.

The generated files contain the SRAM with all the parameters for an actual implementation, including LEF and LIB files and an area report in Figure 27.

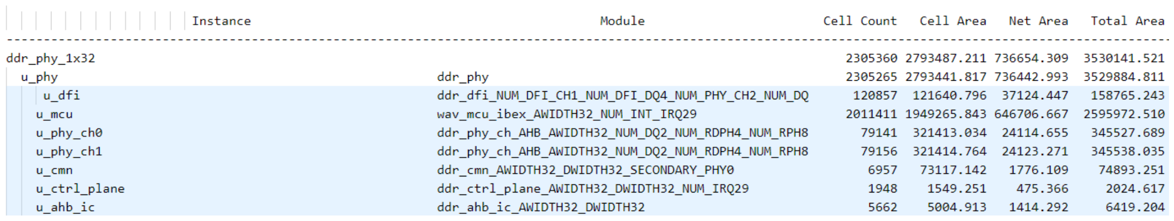


Parameter	Value	Units
Area	14754.460	μm^2
X Dimension	161.096	µm
Y Dimension	91.588	µm
Body Biased P Well Area	3412.984	μm^2
Body Biased N Well Area	2928.009	μm^2

Figure 27 SRAM area report

4. Results

Once the synthesis process finishes it writes multiple reports for the gates that are used, area, power and timing. The one we are most interested in is the area report, as it is the biggest contributor to the cost of the IP. From the area report we get the cell area and the net area, as a design with this complexity will require a significant area for routing. The total area for the DDR PHY is 3.5mm^2 as seen in Figure 28.



```
 Instance                                     Module                                       Cell Count  Cell Area   Net Area   Total Area
----------------------------------------------------------------------------------------------------------------------------------------
ddr_phy_1x32                                                                                  2305360 2793487.211 736654.309  3530141.521
  u_phy                                       ddr_phy                                         2305265 2793441.817 736442.993  3529884.811
     u_dfi                                    ddr_dfi_NUM_DFI_CH1_NUM_DFI_DQ4_NUM_PHY_CH2_NUM_DQ  120857  121640.796  37124.447   158765.243
    u_mcu                                     wav_mcu_ibex_AWIDTH32_NUM_INT_IRQ29             2011411 1949265.843 646706.667  2595972.510
    u_phy_ch0                                 ddr_phy_ch_AHB_AWIDTH32_NUM_DQ2_NUM_RDPH4_NUM_RPH8    79141  321413.034  24114.655   345527.689
    u_phy_ch1                                 ddr_phy_ch_AHB_AWIDTH32_NUM_DQ2_NUM_RDPH4_NUM_RPH8    79156  321414.764  24123.271   345538.035
    u_cmn                                     ddr_cmn_AWIDTH32_DWIDTH32_SECONDARY_PHY0           6957   73117.142   1776.109    74893.251
    u_ctrl_plane                              ddr_ctrl_plane_AWIDTH32_DWIDTH32_NUM_IRQ29         1948    1549.251    475.366     2024.617
    u_ahb_ic                                  ddr_ahb_ic_AWIDTH32_DWIDTH32                       5662    5004.913   1414.292     6419.204
```

Figure 28 final area report

An interesting analysis of this total area is how it is distributed amongst all the internal blocks. As seen in Figure 29, 74% of the total area is taken by the integrated RISC-V controller, this is to be expected as it contains all the logic to control the PHY, also inside this component is the SRAM. Each PHY channel is 10% of the total area, these blocks are mostly analog design, and the area estimation is mostly speculative, and it could change significantly after the real design is finished. The other components like the PLL and the AHB interface have a small percentage of the total area.

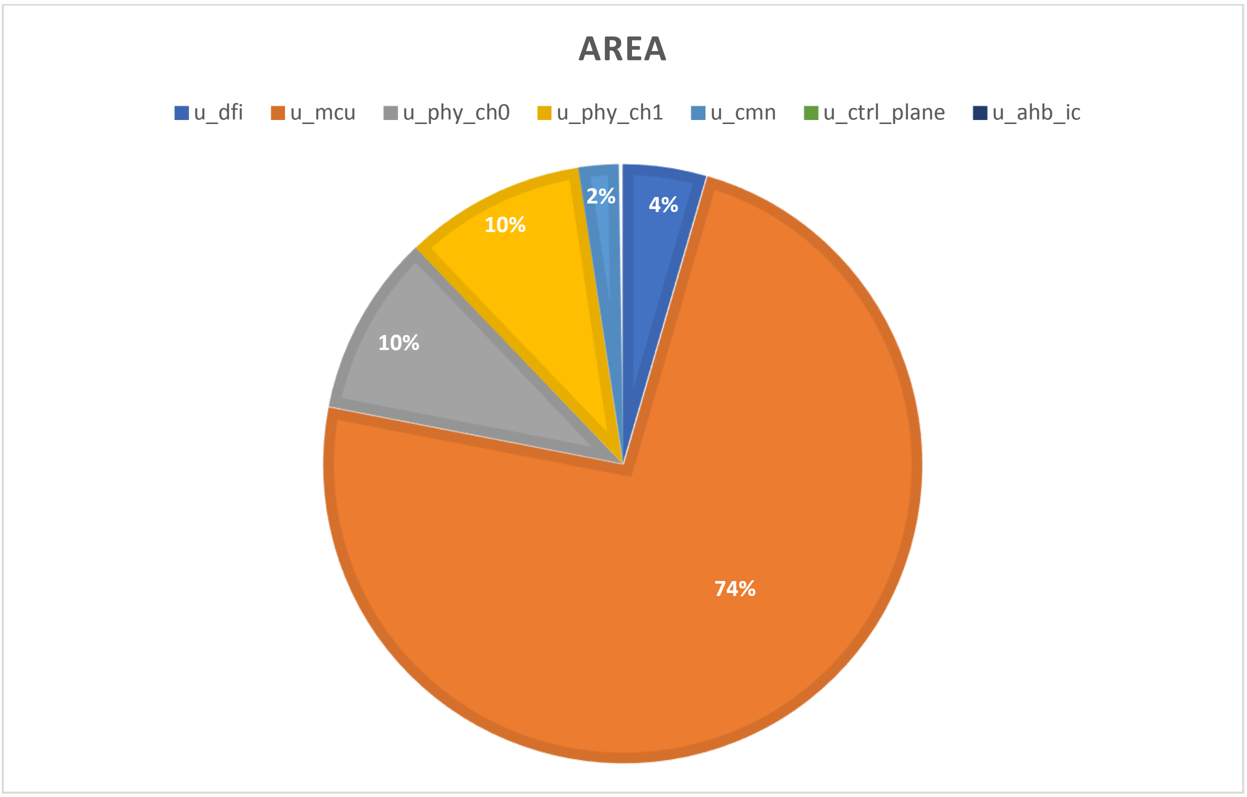
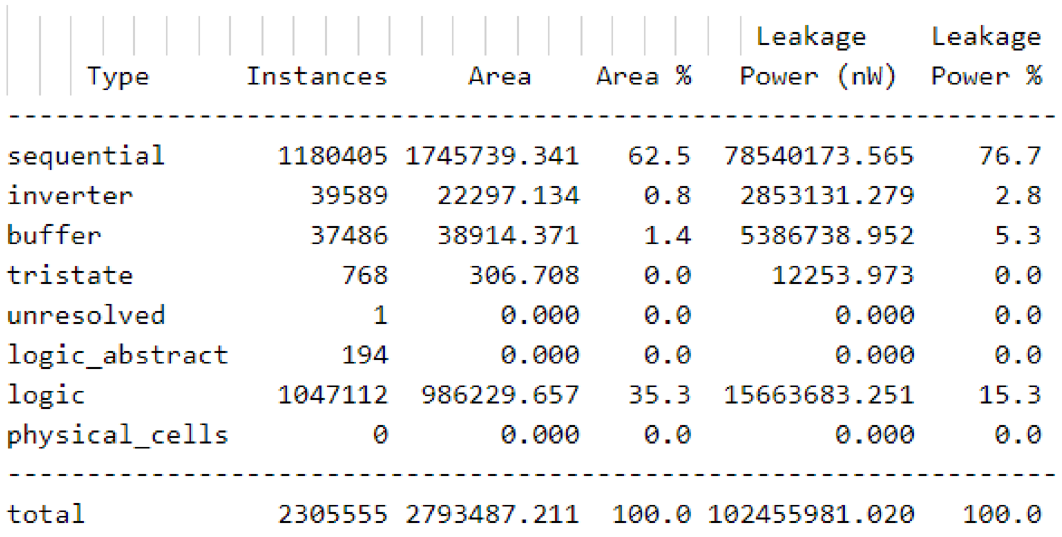


Figure 29 Area distribution

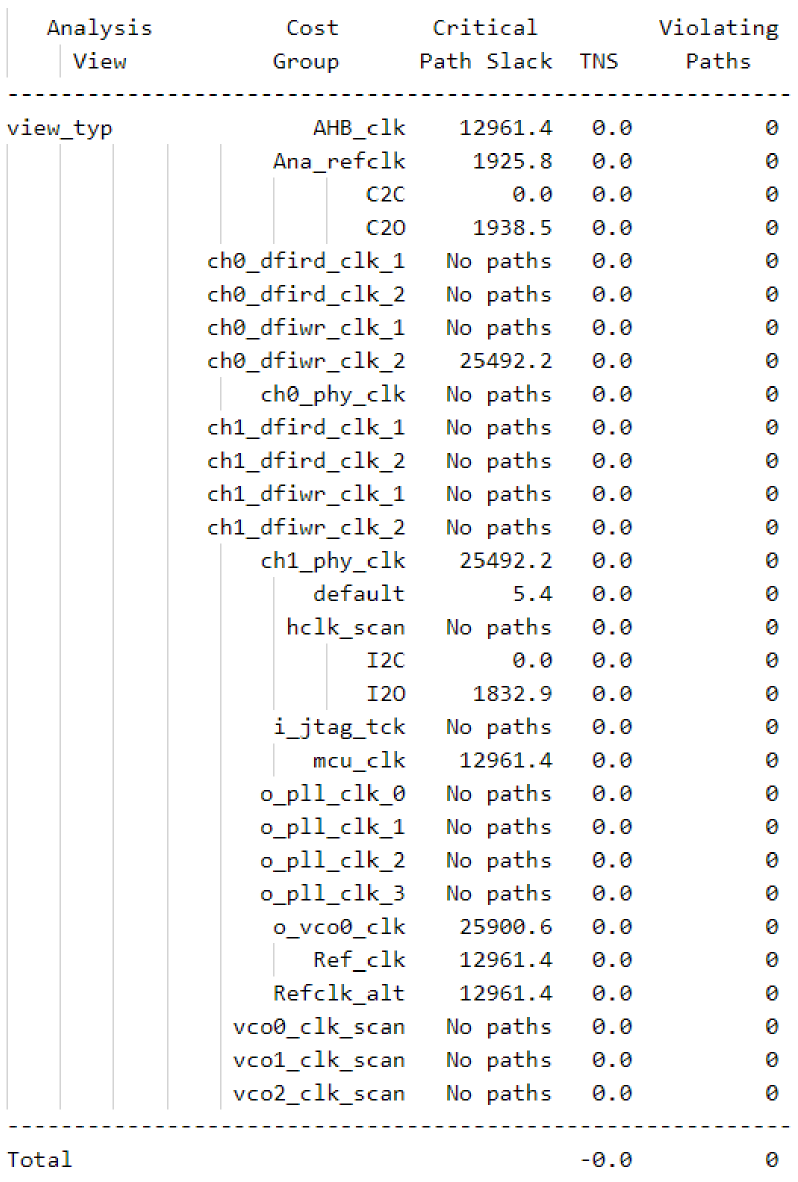
Also, the gates report tells us in detail how many instances of each gate are used in the design and the area contribution of each one. Also, for the cells that are part of the GF22 library contain the leakage power, but the place holders that we generated do not contain this information, so the power consumption report is not accurate at this point in the developement. The total number of cells is 2.305.360 as seen in Figure 30, as we know this is a large and complex design.



```
                                                          Leakage     Leakage
     Type         Instances       Area      Area %    Power (nW)   Power %
-------------------------------------------------------------------
sequential         1180405 1745739.341    62.5   78540173.565     76.7
inverter             39589   22297.134     0.8    2853131.279      2.8
buffer               37486   38914.371     1.4    5386738.952      5.3
tristate               768     306.708     0.0      12253.973      0.0
unresolved               1       0.000     0.0          0.000      0.0
logic_abstract         194       0.000     0.0          0.000      0.0
logic              1047112  986229.657    35.3   15663683.251     15.3
physical_cells           0       0.000     0.0          0.000      0.0
-------------------------------------------------------------------
total              2305555 2793487.211   100.0  102455981.020    100.0
```

Figure 30 Final gates report

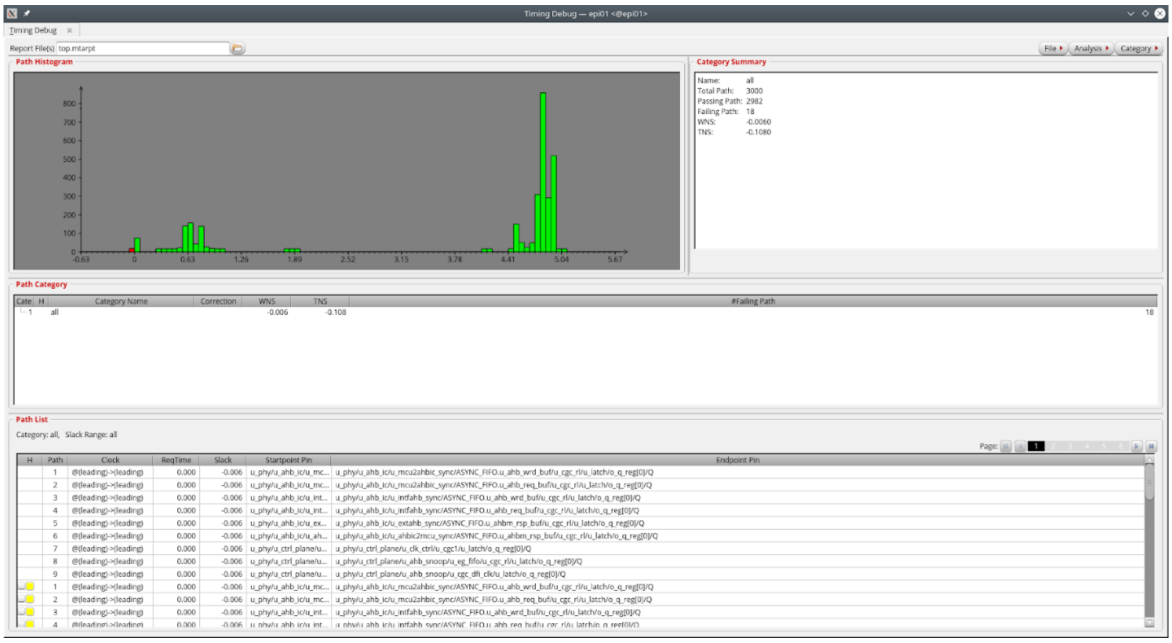
The timing report in Figure 31 gives us information on the slack of each clock line, initially we had doubts about the feasibility of the implementation with 22nm, as the original design was done with 12nm, so there was a risk that 22nm was not fast enough. For now, the synthesis finished with a minimum slack of 0ps, for an actual implementation this would need to be increased, but for now it just proves that the design is feasible with our technology node.



```
  Analysis         Cost       Critical          Violating
    View           Group     Path Slack  TNS      Paths
-----------------------------------------------------------
view_typ              AHB_clk    12961.4   0.0           0
                   Ana_refclk     1925.8   0.0           0
                          C2C        0.0   0.0           0
                          C2O     1938.5   0.0           0
               ch0_dfird_clk_1  No paths   0.0           0
               ch0_dfird_clk_2  No paths   0.0           0
               ch0_dfiwr_clk_1  No paths   0.0           0
               ch0_dfiwr_clk_2   25492.2   0.0           0
                  ch0_phy_clk   No paths   0.0           0
               ch1_dfird_clk_1  No paths   0.0           0
               ch1_dfird_clk_2  No paths   0.0           0
               ch1_dfiwr_clk_1  No paths   0.0           0
               ch1_dfiwr_clk_2  No paths   0.0           0
                  ch1_phy_clk    25492.2   0.0           0
                      default        5.4   0.0           0
                    hclk_scan   No paths   0.0           0
                          I2C        0.0   0.0           0
                          I2O     1832.9   0.0           0
                   i_jtag_tck   No paths   0.0           0
                      mcu_clk    12961.4   0.0           0
                  o_pll_clk_0   No paths   0.0           0
                  o_pll_clk_1   No paths   0.0           0
                  o_pll_clk_2   No paths   0.0           0
                  o_pll_clk_3   No paths   0.0           0
                   o_vco0_clk    25900.6   0.0           0
                      Ref_clk    12961.4   0.0           0
                   Refclk_alt    12961.4   0.0           0
                vco0_clk_scan   No paths   0.0           0
                vco1_clk_scan   No paths   0.0           0
                vco2_clk_scan   No paths   0.0           0
-----------------------------------------------------------
Total                                     -0.0           0
```

Figure 31 Final timing report

The path histogram in Figure 32 provides a view of how the slack is distributed, we are able to see how most of it is within a safe range of 4ns, and then there are a few paths with 600ps of slack, and finally only a few specific paths that are at 0ps, this could be solved by using faster cells for this critical paths, but it is beyond the scope of this thesis.



Report File(s) top.mtarpt

Category Summary

Name:	all
Total Path:	3000
Passing Path:	2982
Failing Path:	18
WNS:	-0.0060
TNS:	-0.1080

Path Category

Cate	H	Category Name	Correction	WNS	TNS	#Failing Path
1		all		-0.006	-0.108	18

Path List

Category: all, Slack Range: all

H	Path	Clock	ReqTime	Slack	Startpoint Pin	Endpoint Pin
	1	@(leading)->(leading)	0.000	-0.006	u_phy/u_ahb_ic/u_mc...	u_phy/u_ahb_ic/u_mcu2ahbic_sync/ASYNC_FIFO.u_ahb_wrd_buf/u_cgc_rl/u_latch/o_q_reg[0]/Q
	2	@(leading)->(leading)	0.000	-0.006	u_phy/u_ahb_ic/u_mc...	u_phy/u_ahb_ic/u_mcu2ahbic_sync/ASYNC_FIFO.u_ahb_req_buf/u_cgc_rl/u_latch/o_q_reg[0]/Q
	3	@(leading)->(leading)	0.000	-0.006	u_phy/u_ahb_ic/u_int...	u_phy/u_ahb_ic/u_intfahb_sync/ASYNC_FIFO.u_ahb_wrd_buf/u_cgc_rl/u_latch/o_q_reg[0]/Q
	4	@(leading)->(leading)	0.000	-0.006	u_phy/u_ahb_ic/u_int...	u_phy/u_ahb_ic/u_intfahb_sync/ASYNC_FIFO.u_ahb_req_buf/u_cgc_rl/u_latch/o_q_reg[0]/Q
	5	@(leading)->(leading)	0.000	-0.006	u_phy/u_ahb_ic/u_ex...	u_phy/u_ahb_ic/u_extahb_sync/ASYNC_FIFO.u_ahbm_rsp_buf/u_cgc_rl/u_latch/o_q_reg[0]/Q
	6	@(leading)->(leading)	0.000	-0.006	u_phy/u_ahb_ic/u_ah...	u_phy/u_ahb_ic/u_ahbic2mcu_sync/ASYNC_FIFO.u_ahbm_rsp_buf/u_cgc_rl/u_latch/o_q_reg[0]/Q
	7	@(leading)->(leading)	0.000	-0.006	u_phy/u_ctrl_plane/u...	u_phy/u_ctrl_plane/u_clk_ctrl/u_cgc1/u_latch/o_q_reg[0]/Q
	8	@(leading)->(leading)	0.000	-0.006	u_phy/u_ctrl_plane/u...	u_phy/u_ctrl_plane/u_ahb_snoop/u_eg_fifo/u_cgc_rl/u_latch/o_q_reg[0]/Q
	9	@(leading)->(leading)	0.000	-0.006	u_phy/u_ctrl_plane/u...	u_phy/u_ctrl_plane/u_ahb_snoop/u_cgc_dfi_clk/u_latch/o_q_reg[0]/Q
🟨	1	@(leading)->(leading)	0.000	-0.006	u_phy/u_ahb_ic/u_mc...	u_phy/u_ahb_ic/u_mcu2ahbic_sync/ASYNC_FIFO.u_ahb_wrd_buf/u_cgc_rl/u_latch/o_q_reg[0]/Q
🟨	2	@(leading)->(leading)	0.000	-0.006	u_phy/u_ahb_ic/u_mc...	u_phy/u_ahb_ic/u_mcu2ahbic_sync/ASYNC_FIFO.u_ahb_req_buf/u_cgc_rl/u_latch/o_q_reg[0]/Q
🟨	3	@(leading)->(leading)	0.000	-0.006	u_phy/u_ahb_ic/u_int...	u_phy/u_ahb_ic/u_intfahb_sync/ASYNC_FIFO.u_ahb_wrd_buf/u_cgc_rl/u_latch/o_q_reg[0]/Q
🟨	4	@(leading)->(leading)	0.000	-0.006	u_phy/u_ahb_ic/u_int...	u_phy/u_ahb_ic/u_intfahb_sync/ASYNC_FIFO.u_ahb_req_buf/u_cgc_rl/u_latch/o_q_reg[0]/Q

Figure 32 Slack distribution

Finally a picture of the synthesised design in Figure 33, it is hard to identify all the blocks and the connectivity, because the design is so complex, but this schematic tool can be useful to identify problematic paths and better understand their connectivity and how they are composed.

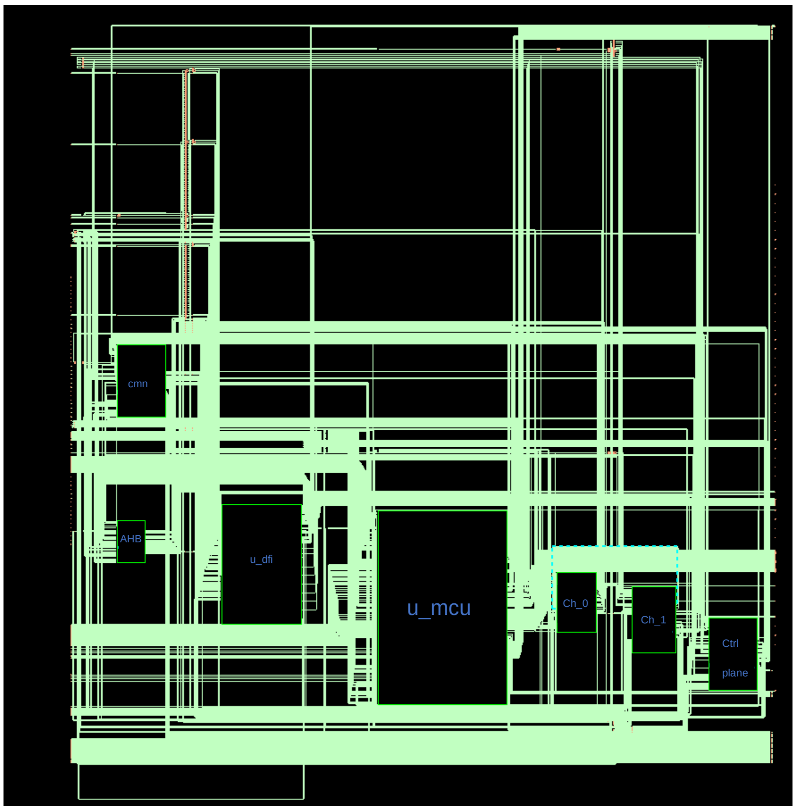


Figure 33 GUI representation

5. Budget

As there is no prototype in this project, the costs are the engineer's work and the software licenses used, mainly. The project started on 1th February and ended to 1th July, resulting in approximately 110 work days. Assuming 4h per day (440 h) and a salary of 9 €/h we get 3960 €.

The Cadence license should be added too. Cadence is not revealing their prices for their products, and it is not possible to know the cost of this license. However, we used an academic Cadence license and these prices are known.

The software used needs every year 600€ for "software only membership europractice" and 1890€ for "Cadence IC package (1-5 lic)". Since we used the software during 6 months, we will consider half the price.

For a total of 5200€.

6. Conclusions and future development

The scope of this thesis was to study the Wavious open source IP for the DDR PHY, this proved to be challenging as the documentation on the analog and mixed-signal are quite limited at this point in time, although the company promised to release further datasheets on these components.

So, the future developments will consist of designing these components, this design will require validation on the functionality and the robustness against temperature and manufacturing variance.

Also, the synthesis process has been challenging as there's limited data on clock structure, therefore the synthesis still reports some issues with the timing specifications. Once this stage of the IC design flow is finished with satisfactory result there is still the place and route stage and the validation process.

Finally, this is only the design of the DDR PHY component, the full design of a RISC-V ASIC that has direct access to a RAM memory will take the effort of multiple people over years.

7. Bibliography

[1] DRAC project [Online]. Available: https://drac.bsc.es/

[2] “Wavious,” [Online]. Available: http://www.wavious.com/.

[3] C. Kim, *High-Bandwidth Memory Interface Design,* Dept. of Electrical Engineering Korea University, Seoul, Korea, 2013.

[4] J. P. Kulkarni, K. Kim and K. Roy, “A 160 mV Robust Schmitt Trigger Based Subthreshold SRAM,” *IEEE Journal of Solid-State Circuits,* no. doi: 10.1109/JSSC.2007.897148., 2007.

[5] M. A. Islam, M. Y. Arafath and M. J. Hasan, "Design of DDR4 SDRAM controller," 8th International Conference on Electrical and Computer Engineering, 2014, pp. 148-151, doi: 10.1109/ICECE.2014.7026950.

[6] K.-H. Koo, *Versatile IO Circuit Schemes for LPDDR4 with 1.8mW/Gbps/pin Power Efficiency,* Santa Clara, CA: UBM Tech, 2014.

[7] JEDEC SOLID STATE TECHNOLOGY ASSOCIATION, *JEDEC STANDARD, Low Power Double Data Rate 4 (LPDDR4),* JESD209-4, 2014.

[8] Marc Greenberg, *Architectural Options for LPDDR4 Implementation in Your Next Chip Design,* JEDEC Mobile & IOT Forum, 2016.

[9] Cadence, *Genus Basic Rapid Adoption Kit,* 2015.

[10] WDDR, 2021. [Online]. Available: https://github.com/waviousllc/wav-lpddr-hw.git.

[13] B. Xiang, Y. Fan, J. Ayers, J. Shen and D. Zhang, “A 0.5V-to-0.9V 0.2GHz-to-5GHz Ultra-Low-Power Digitally-Assisted Analog Ring PLL with Less Than 200ns Lock Time in 22nm FinFET CMOS Technology,” *2020 IEEE Custom Integrated Circuits Conference,* no. doi: 10.1109/CICC48029.2020.9075897., 2020.

[14] A. Stefanou and E. Bochoridis, “Design of a Low-Power Phase Interpolator for Multi-Standard Transceiver PHYs,” *2019 8th International Conference on Modern Circuits and Systems Technologies (MOCAST),* no. doi: 10.1109/MOCAST.2019.8742046., 2019.

[15] Radhika, N. Pandey, K. Gupta and M. Gupta, “Low power D-latch design using MCML tri-state buffers,” *2014 International Conference on Signal Processing and Integrated Networks (SPIN),* no. doi: 10.1109/SPIN.2014.6777011., 2014.

[16] G.-M. HONG, “A LPDDR4 MEMORY CONTROLLER DESIGN WITH EYE CENTER DETECTION ALGORITHM,” *PH.D. DISSERTATION, DEPARTMENT OF ELECTRICAL ENGINEERING AND COMPUTER SCIENCE COLLEGE OF ENGINEERING SEOUL NATIONAL UNIVERSITY,* .

[17] Cadence, “Basics of Analog Flow: A Design Oriented Approach,”.

[18] Cadence, “Abstract Generator Flow (GUI/Batch mode),”.

8. Appendices

The following documents are uploaded in the Intranet:

final_qor.rpt

final_area.rpt

final_time_view_typ.rpt

final_gates.rpt

Glossary

ASIC:	Application-Specific Integrated Circuit
FPGA:	Field Programmable Gate Array
BSC:	Barcelona Supercomputing Center
RAM:	Random Access Memory
DDR:	Double Data Rate
PHY:	Physical Layer
RISC-V:	Reduced Instruction Set Computing five
IC:	Integrated Chip
IP:	Intellectual Property
SDC:	Synopsys Design Constraint
CGC:	Clock Gating Cell
TCL:	Tool Command Language
PLL:	Phase Locked Loop
FDSOI:	Fully Depleted Silicon On Insulator
LEF:	Library Exchange Format
LIB:	Library
SRAM:	Static RAM
DRAM:	Dynamic RAM
LVSTL:	Low Voltage Swing Terminated Logic
GF:	Global Foundries
RTL:	Register-Transfer Level
PoP:	Package on Package