Cologne Chip

GateMate™ FPGA Datasheet

CCGM1A1
CCGM1A2

Cologne Chip AG
Eintrachtstr. 113
50668 Köln

Tel.: +49 (0) 221 / 91 24-0
Fax: +49 (0) 221 / 91 24-100

https://colognechip.com
info@colognechip.com

Contents

List of Figures

List of Tables

About this Document

This datasheet covers all features of the Cologne Chip GateMate™ FPGA Series and is part of the GateMate™ documentation collection.

File ds1001-gatemate1-attachment-latest.zip is published together with this document. It contains the pin lists of CCGM1A1 and CCGM1A2 in csv[1] file format.

For more information please refer to the following documents:

- Technology Brief of GateMate™ FPGA
- **DS1002** – GateMate™ FPGA Programmer Board Datasheet
- **DS1003** – GateMate™ FPGA Evaluation Board Datasheet
- **UG1001** – GateMate™ FPGA Primitives Library
- **UG1002** – GateMate™ FPGA Toolchain Installation User Guide

Cologne Chip provides a comprehensive technical support. Please visit our website for more information or contact our support team.

[1] csv: comma-separated values

Revision History

This datasheet is constantly updated. The latest version of the document can be found following the link below:

DS1001 – GateMate™ FPGA CCGM1A1 / CCGM1A2 Datasheet

Date	Remarks
February 2025	• SerDes interface description added in Section 2.5 from page 50. • JTAG instructions `JTAG_WR_SERDES_REGFILE` and `JTAG_WR_SERDES_REGFILE` added in Section 3.4.7 from page 153. • SerDes electrical characteristics added in Table 4.3 on page 159.
May 2024	• CCGM1A2 added to the datasheet content. See Section 1.2 from page 20 for features of both chips CCGM1A1 / CCGM1A2 and Section 1.3 for the flip-chip and multi-die concept. • The description of the clock generators has been greatly expanded and deepened in Section 2.6 from page 96. • Routing concept of the die-to-die connections added in Section 2.9.2 on page 124. • With this document, additional files with pin lists for CCGM1A1 and CCGM1A2 are also available for the first time in csv file format in the Cologne Chip download area.
February 2024	The GateMate™ FPGA is now qualified for the extended temperature range from –40 °C to +125 °C (see Table 4.2 on page 158).
December 2023	Internal pull-up / pull-down resistors of `POR_EN` and `RST_N` corrected in Figures 3.2, 3.4 and 3.5 with corresponding changes in the text of Sections 3.2.2 and 3.2.3 from page 136.
November 2023	Information added in Section 4.
January 2023	Minor changes in Section 3.1.
…	
February 2020	Initial pre-release.

General Remarks to Notations

1. The decimal point is written as a point (e.g., 1.23). Thousands separators are written with thin space.
2. Numerical values have different notations for various number systems; e.g., the hexadecimal value `0x C9` is `0b 1100 1001` in binary and 201 in decimal notation.
3. The prefix 'kilo' is written k for the meaning of 1000 and it is written K for the meaning of 1024.
4. Functional elements of the GateMate™ FPGA have certain colors. These are:

   - Cologne Programmable Element (CPE)
   - Input Multiplexer (IM)
   - Output Multiplexer (OM)
   - Small Switch Box (SB)
   - Big Switch Box (SB)
   - General purpose input / output (GPIO)
   - Dual port SRAM (DPSRAM)
   - Phase-locked loop (PLL)
   - Serializer / deserializer (SerDes)

5. General purpose input / output (GPIO) banks and special pin functions of the GateMate™ FPGA are grouped by certain colors. Please see Table 5.2 on page 169 or Figure 5.1 on page 166 for an overview.

Chapter 1

Introduction

1.1 Introduction

The Cologne Chip GateMate™ FPGA Series is a family of small to medium-sized FPGAs that addresses all requirements of typical applications. The programmable silicon can be used from low power to high speed applications and thus in a wide range of fields: industry, automation, communication, security, automotive, Internet of Things (IoT), artificial intelligence (AI), and lots more.

Logic capacity, power consumption, package size and printed circuit board (PCB) compatibility are optimized for a wide range of applications. As these FPGAs combine various features with lowest cost in the market, the devices are well suited from university projects to high volume applications. Because of the outstanding ratio of circuit size to cost, even price sensitive applications can also now take advantage of FPGAs. The devices are manufactured using Globalfoundries™ 28 nm Super Low Power (SLP) process in Dresden, Germany. The GateMate™ FPGA program is supported by the German Federal Ministry for Economic Affairs and Energy as part of the Important Project of Common European Interest (IPCEI) on Microelectronics project, which is also supported by the European Commission.

Supported by:

Federal Ministry
for Economic Affairs
and Energy

on the basis of a decision
by the German Bundestag

Cologne Chip is a semiconductor company based in Cologne, Germany. The company, which celebrates its 25th anniversary in 2020, has excellent industry knowledge and an experienced team of developers. The design and manufacturing location *Made in Germany* represents a unique selling proposition of globally competitive FPGAs in the market.

Cologne Chip offers and supports a variety of development tools:

- The GateMate™ FPGA Evaluation Board is a feature-rich, ready-to-use development platform that serves as a reference design and for a direct entry into application development.
- With the help of the GateMate™ FPGA Programmer, the FPGA can be configured in various ways.

Cologne Chip provides a comprehensive technical support. Please visit our website for more information or contact our support and sales teams.

1.2 Features

This datasheet covers two GateMate™ FPGAs, namely

CCGM1A1 which is the basic GateMate™ FPGA and

CCGM1A2 which consits of two CCGM1A1 dies in one package.

Between the two dies of CCGM1A2, die-to-die connections link the routing structures together.

Novel programmable element architecture

- Each die has
  - 20,480 programmable elements for combinatorial and sequential logic
  - 20,480 8-input LUT-trees / 40,960 4-input LUT-trees
  - 40,960 latches / flip-flops within programmable elements
- Each programmable element configurable as:
  - 1-bit full adder
  - 2-bit full adder or
  - 2×2-bit multiplier
- Dedicated logic and routing for fast arithmetic and arbitrary-sized multipliers

9 general purpose input / output (GPIO) banks

- 162 user-configurable GPIOs
- All GPIOs configurable as single-ended or LVDS differential pairs
- Double data rate (DDR) registers in I/O cells
- All GPIOs MIPI D-PHY compatible (CSI-2)
- I/O voltage range from 1.2 to 2.5 V

Clock generators (PLLs)

- Each die has 4 clock generators
- Maximum oscillator frequency from 1,000 MHz to 2,500 MHz

5.0 Gb/s serializer / deserializer (SerDes) controller

- Each die has 1 SerDes interface

Flexible memory resources

- Each die has 1,310,720 total RAM bits distributed over 32 SRAM blocks
- Each RAM block configurable as two independent 20 Kbit blocks or single 40 Kbit block
- Simple or True Dual Port (SDP / TDP) or FIFO mode
- Data width from 1 bit up to 40 bits (TDP) or 80 bits (SDP)
- Bit-wide write enable
- Error checking and correcting (ECC) for certain bit widths

Flexible device configuration

- Configuration bank switchable to user I/O
- JTAG interface with bypass to SPI interface
- Active / passive SPI interface for singe-, dual- and quad-mode operation
- SPI flash interface in active mode
- Multi-chip configuration from single source

Performance modes: *low power, economy, speed*

- Mode adjustable on each chip via core voltage
- Core voltage range from 0.9 to 1.1 V

Package

- 15×15 mm 324 balls 0.8 mm fine pitch Ball Grid Array (FBGA) package
- Only 2 signal layers required on PCB

Overview of the CCGM1A1 und CCGM1A2 differences:

	rel.	Cologne Programmable Elements		Block RAMs [*3]			
Device	**size**	**CPEs [*1] [*2]**	**FF / Latches**	**20 Kb**	**40 Kb**	**PLLs**	**SerDes**
CCGM1A1	1	20,480	40,960	64	32	4	1
CCGM1A2	2	40,960	81,920	128	64	8	2

[*1] CPEs have 2 x 4 or 8 inputs connected to a LUT-tree
[*2] Each CPE can be used as a 2 x 2 multiplier tile
[*3] Block RAM can have a data width of 1 . . 80 bits

	GPIOs		Package	
Device	**single-ended**	**differential**	**balls**	**size [mm]**
CCGM1A1	162	81	324 BGA	15 x 15
CCGM1A2	162	81	324 BGA	15 x 15

1.3 Flip-Chip and Multi-Die Concept

The GateMate FPGA dies are designed as so-called *flip-chips*. In the flip-chip process, the die is placed directly on the interposer with the active area at the bottom, without any further connecting bond wires. The electrical connection is created by bumps which are applied to the bond pads of the silicon. Figure 1.1 illustrates this procedure and shows how the BGA pins are arranged in the process.

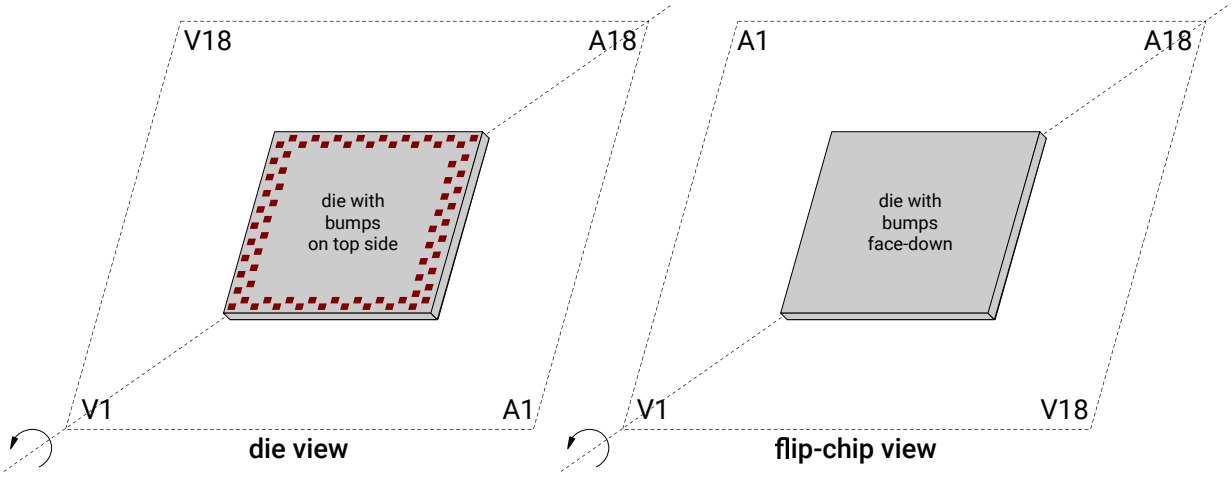


Figure 1.1: *Flip-chip concept for CCGM1A1*

A key advantage of using a flip-chip package is enhanced integration and performance. Flip-chip packaging typically results in lower parasitic effects such as inductance and capacitance, leading to improved signal integrity at high frequencies.

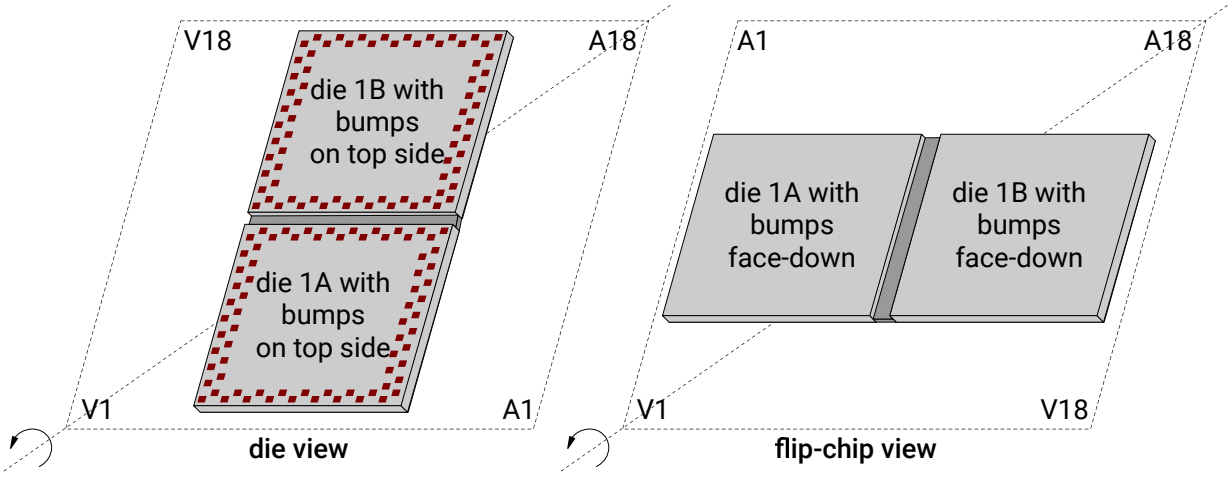


Figure 1.2: *Flip-chip concept for CCGM1A2*

The CCGM1A2 contains two dies of the CCGM1A1, which are interconnected by so-called *die-to-die connections* directly on the silicon. Figure 1.2 shows the arrangement inside the BGA package.

Chapter 2

Architecture

2.1 GateMate™ FPGA Overview

The basic functional elements of GateMate™ FPGA are set up in an array structure of 160 × 128 elements called Cologne Programmable Element (CPE) as described in Section 2.2.

All CPEs are interconnected by a routing structure of 132 × 164 so-called Switch Boxes (SBs) as described in Section 2.9.

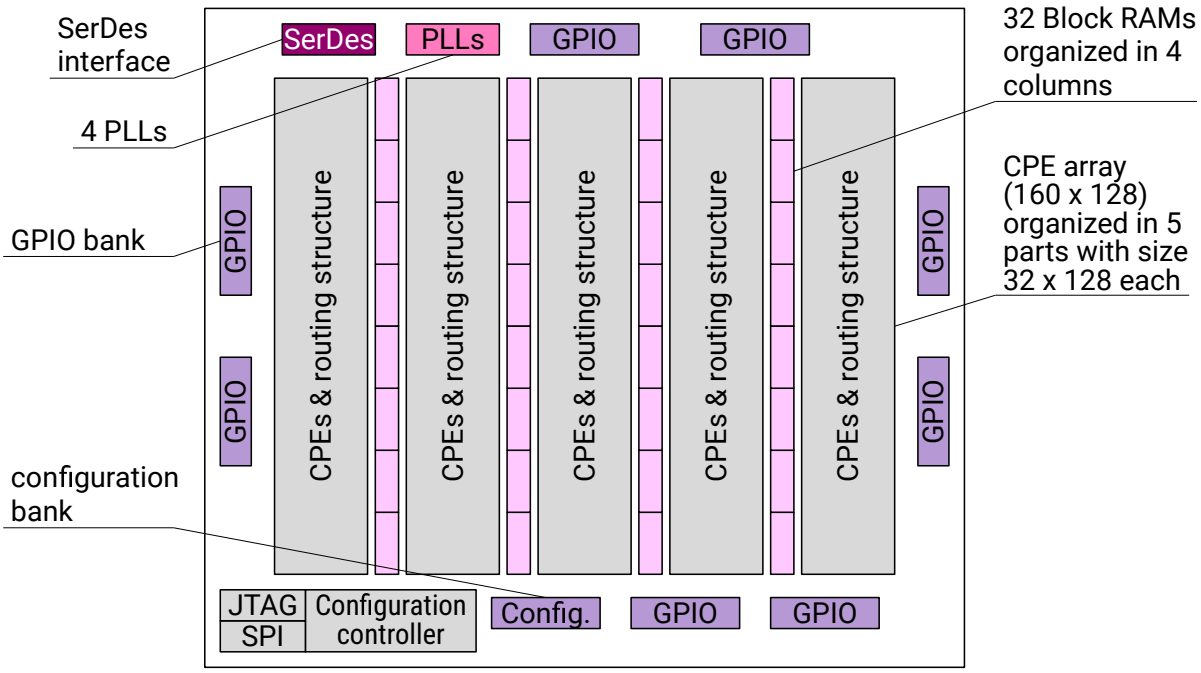


Figure 2.1: *Simplified architecture overview of CCGM1A1 (die view)*

Additional functional blocks are dual port SRAMs (DPSRAMs), phase-locked loops (PLLs), general purpose input / output (GPIO) cells, SPI configuration and data flash interface, JTAG interface and serializer / deserializer (SerDes) interface.

Figure 2.1 shows the architecture of CCGM1A1. The multi-die architecture of CCGM1A2 with two CCGM1A1 dies in one package is shown in Figure 2.2.

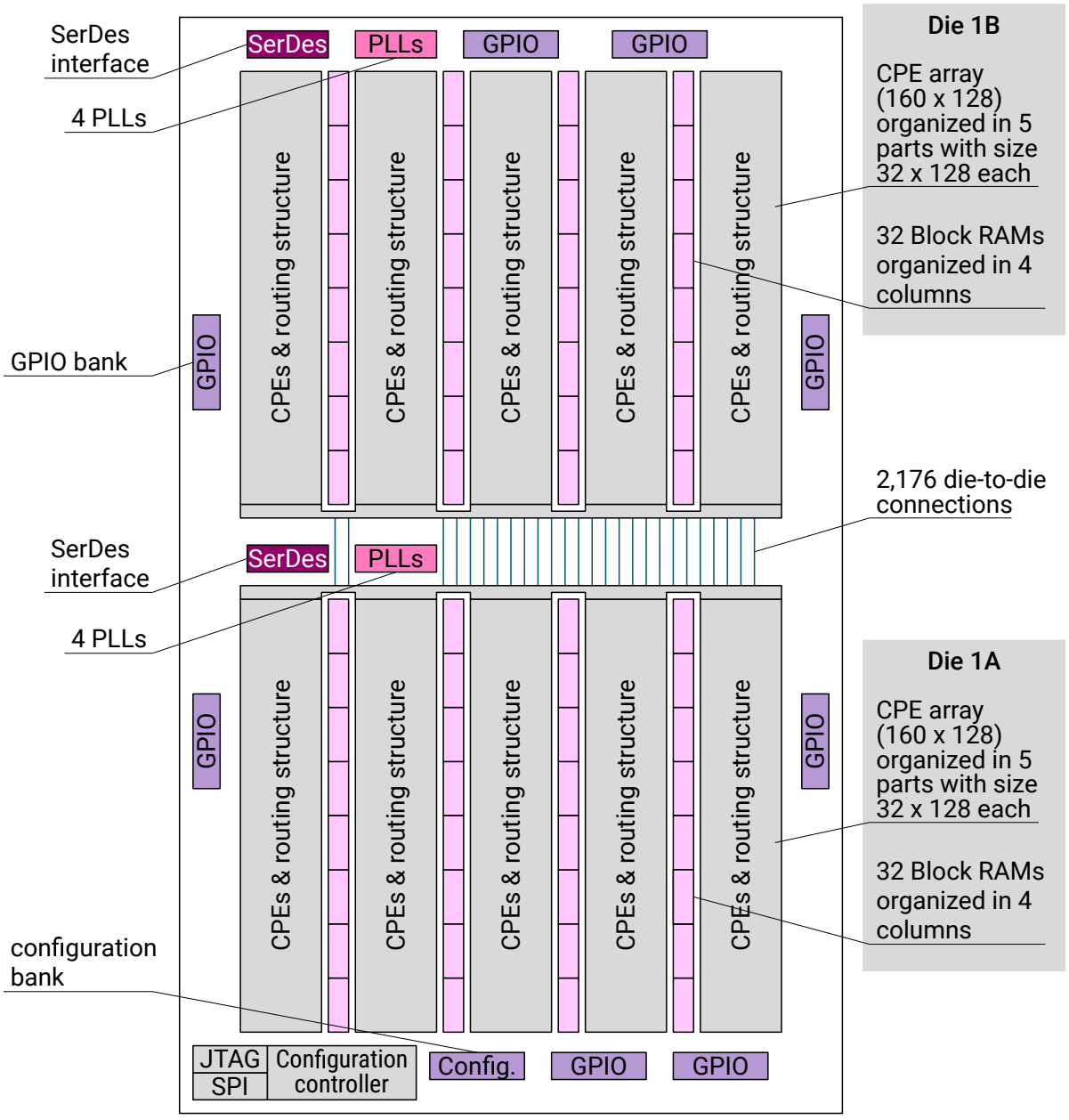


Figure 2.2: *Simplified architecture overview of CCGM1A2 (die view)*

2.2 Cologne Programmable Element (CPE)

General purpose combinatorial and sequential circuits are implemented using CPEs. The CCGM1A1 has 20,480 CPEs arranged in a 160 × 128 matrix. The CCGM1A2 has 40,960 CPEs arranged in two 160 × 128 matrices.

Each CPE can be set up to the following combinatorial functions:

- Dual 4 inputs with 2-input lookup table (LUT-2) tree each
- 8 inputs with LUT-2 tree
- 6 inputs with 4-input multiplexer (MUX-4) function
- 1-bit or 2-bit full adder, expandable to any length in horizontal or vertical arrangement
- 2 × 2-bit multiplier, expandable to any multiplier size

Figure 2.3 illustrates the general structure of a CPE. General combinatorial and sequential functions are supported from fast signal path routing. In addition, the DPSRAM blocks of GateMate™ FPGA, as described in Chapter 2.4, require some CPE ports `RAM_I[2:1]` and `RAM_O[2:1]` for CPE-to-SRAM connection.

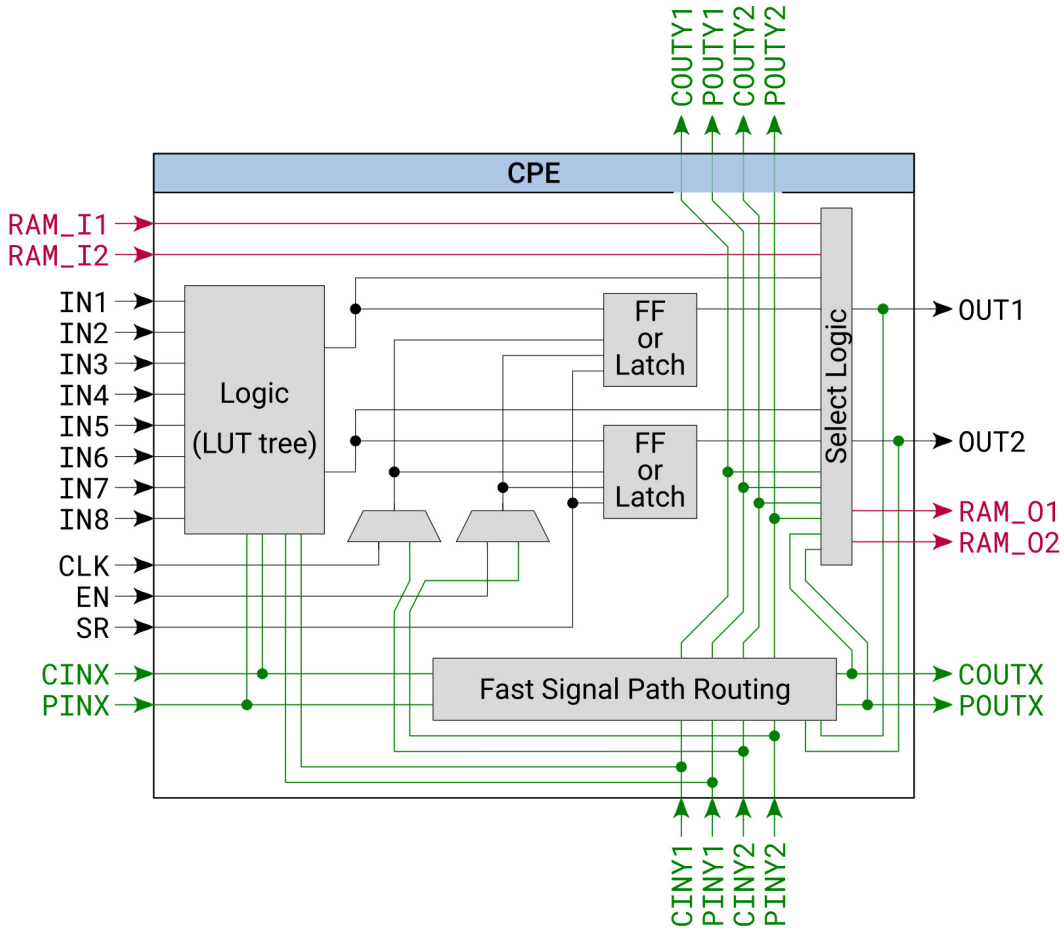


Figure 2.3: *Cologne Programmable Element (CPE)*

Two CPE outputs OUT1 and OUT2 are available and each can use a Flip-Flop or Latch function.

Furthermore, the CPEs include fast signal routing paths, so-called carry & propagation signal lines (CP-lines). Typically, they are used for fast clock and carry propagation in adder and multiplier functions, e.g., fast CPE to adjacent CPE connections, or even fast signal propagation over any span of CPEs.

2.3 General purpose input / output (GPIO)

The GateMate™ FPGAs CCGM1A1 / CCGM1A2 offer up to 162 general purpose input / output pins (GPIOs). These are organized in signal pairs, so-called *pad cells*, which can either operate as two independent, bidirectional single-ended signals or they can be set up as bidirectional low-voltage differential signaling (LVDS). Figures 2.4 and 2.5 show the structure of the pad cells and their interconnection to the FPGA elements.

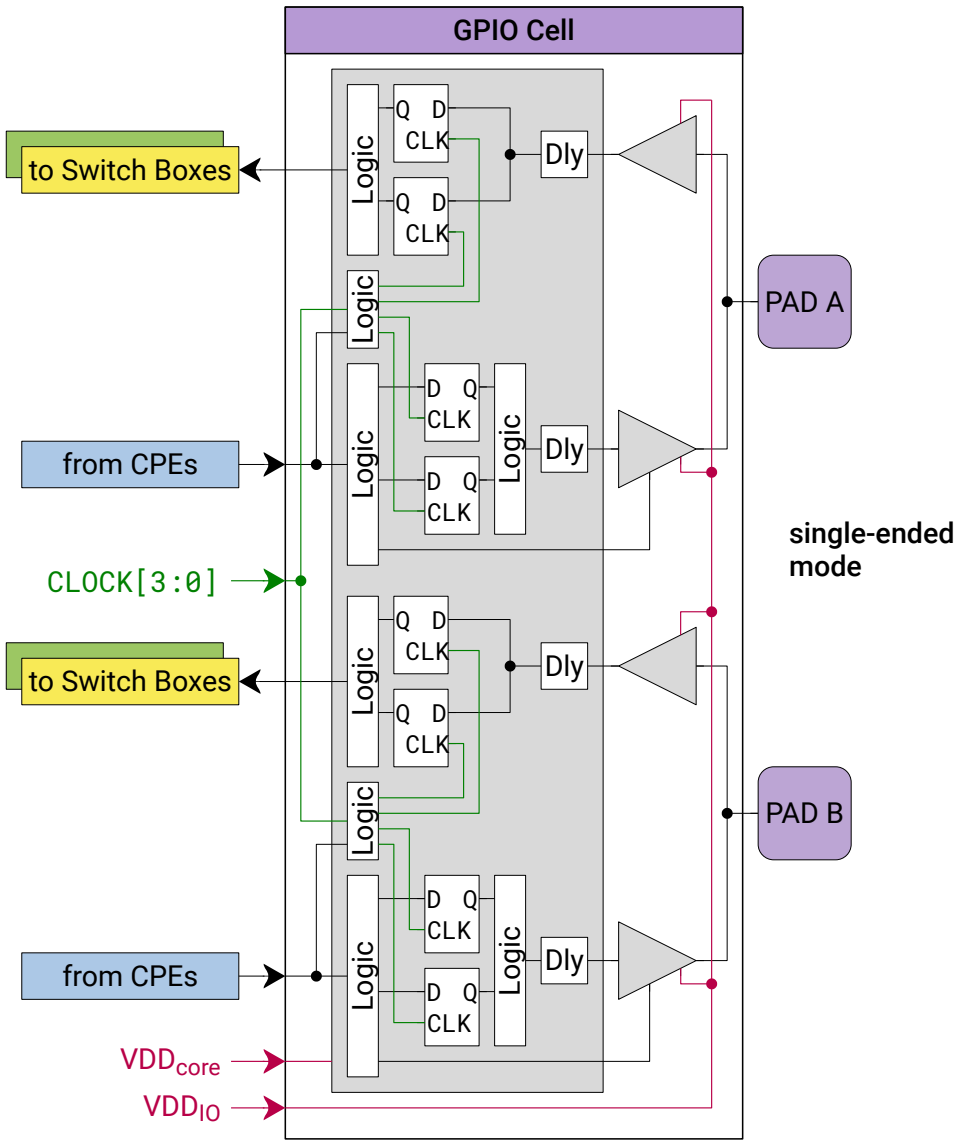


Figure 2.4: *GPIO cell in single-ended mode*

Nine pad cells build up a GPIO bank as shown in Figure 2.6 on page 29. Eight banks and another optional bank [1] are available.

[1] The FPGA configuration pins are arranged within the ninth GPIO bank. These pins can be used as normal GPIO bank after configuration process.

The single-ended GPIO support the low voltage CMOS (LVCMOS) standard up to 2.5 V nominal supply voltage compliant to the standard JESD8-7(A), which means that the GPIO voltage can be 1.8 V ±0.15 V (normal range) or 1.2 V . . 1.95 V (wide range).

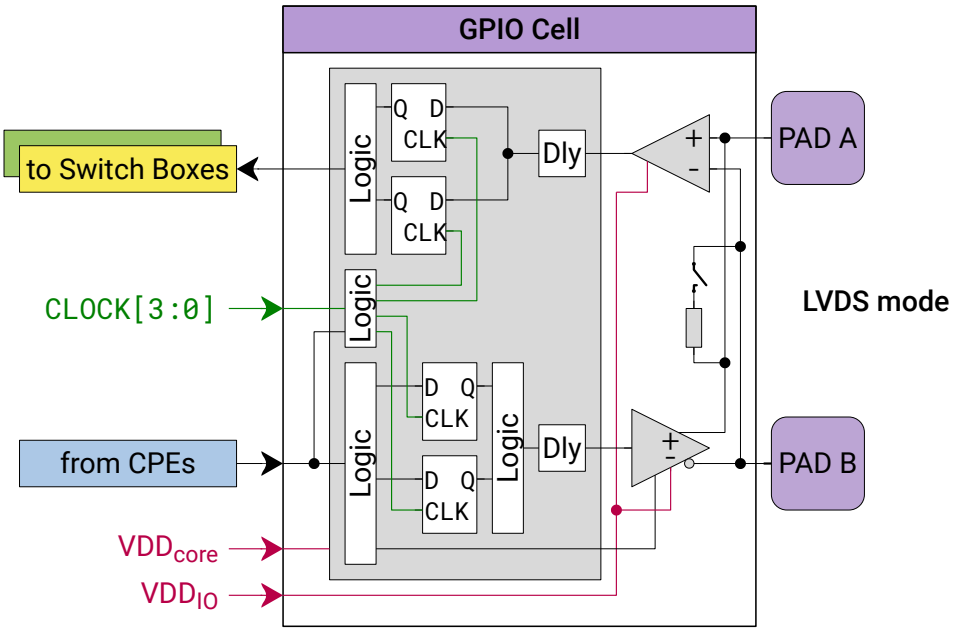


Figure 2.5: *GPIO cell in LVDS mode*

The single-ended GPIO pins have the following output features:

- 0 / 1 / high-Z
- Slew-rate control
- Driver strength 3 mA, 6 mA, 9 mA and 12 mA
- 2 Flip-Flops for SDR and DDR operation
- Programmable delay line

The single-ended GPIO pins have the following input features:

- Schmitt-trigger input
- Pull-up / pull-down resistors or keeper functionality
- Input receiver disable for power reduction
- 2 Flip-Flops for SDR and DDR operation
- Programmable delay line

The LVDS GPIO pins have the following output features:

- Driver strength 3.2 mA and 6.4 mA
- 2 Flip-Flops for SDR and DDR operation
- Programmable delay line

The LVDS GPIO pins have the following input features:

- 2 Flip-Flops for SDR and DDR operation
- Programmable delay line

If a pad cell is used in its LVDS mode, both pads can be used together only. The LVDS pad is compliant to the LVDS 2.5 V standard. It further can operate down to 1.8 V nominal supply voltage, with common mode voltage $V_{DD}/2$. The LVDS input receiver can be used as voltage comparator.

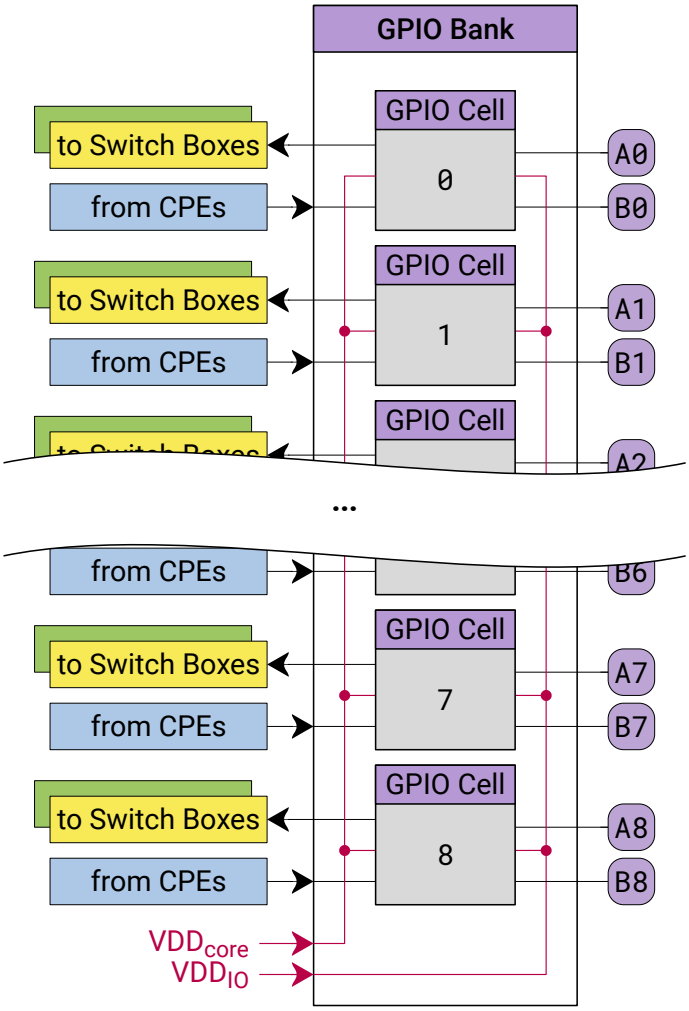


Figure 2.6: *GPIO bank*

> **Please note!**
>
> Unused GPIO banks should still be supplied with power to avoid interference. Any GPIO voltage available on the printed circuit board (PCB) can be used for this purpose.

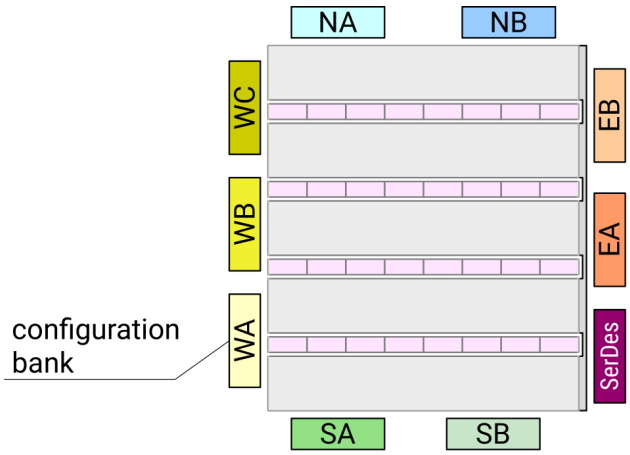


Figure 2.7: *GPIO banks of CCGM1A1 (chip view)*

There are two GPIO banks on each edge of the CCGM1A1 / CCGM1A2 named after the cardinal directions north (N), west (W), east (E) and south (S). As an exception, a third bank for configuration signals is attached to the west edge as shown in Figure 2.7.

For the CCGM1A2, some GPIO banks of the two dies are not connected to the BGA package balls due to pin compatibility with the CCGM1A1. As shown in Figure 2.8, pairs of GPIO banks are connected together. This increases the number of available GPIOs , but it must be noted that of course only one signal of each pair can be enabled as output driver.

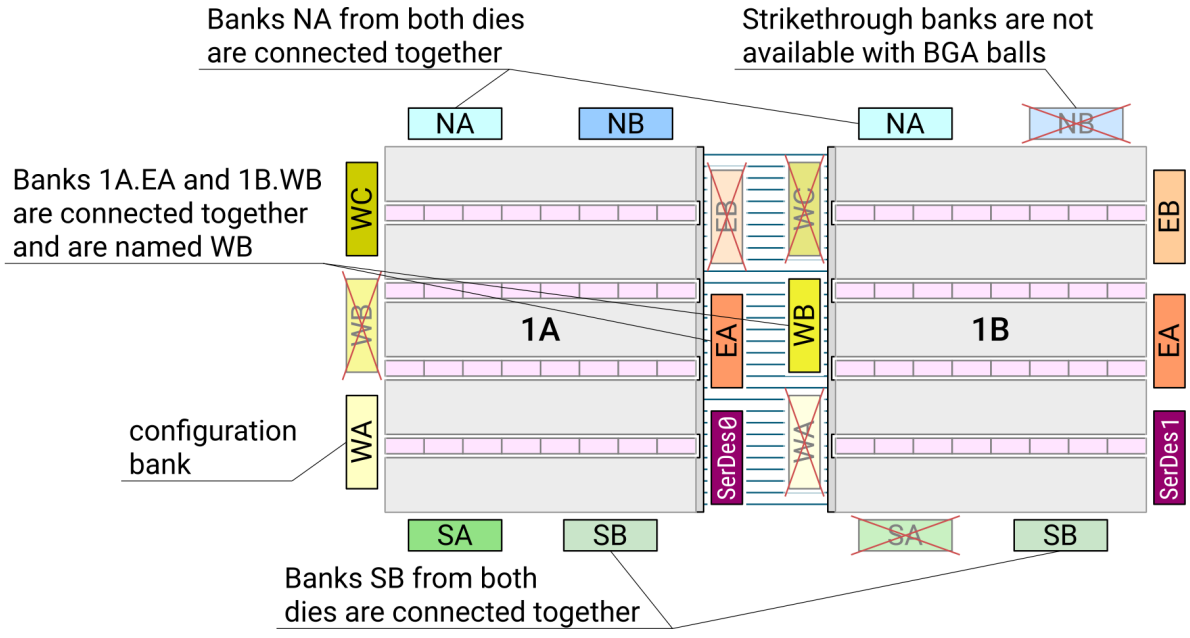


Figure 2.8: *GPIO banks of CCGM1A2 (chip view)*

2.4 Embedded Block RAM

2.4.1 Block RAM Overview

The GateMate™ FPGAs contains a number of embedded volatile memory cells which are organized as configurable 40 kbit dual port SRAM (DPSRAM) cells. These random-access memory (RAM) cells are arranged in columns between the CPE array and routing structure as illustrated in Figure 2.9. Block RAMs have an address (x, y) with $x = 0..3$ and $y = 0..7$. The total number of Block RAM resources is listed in Table 2.1.

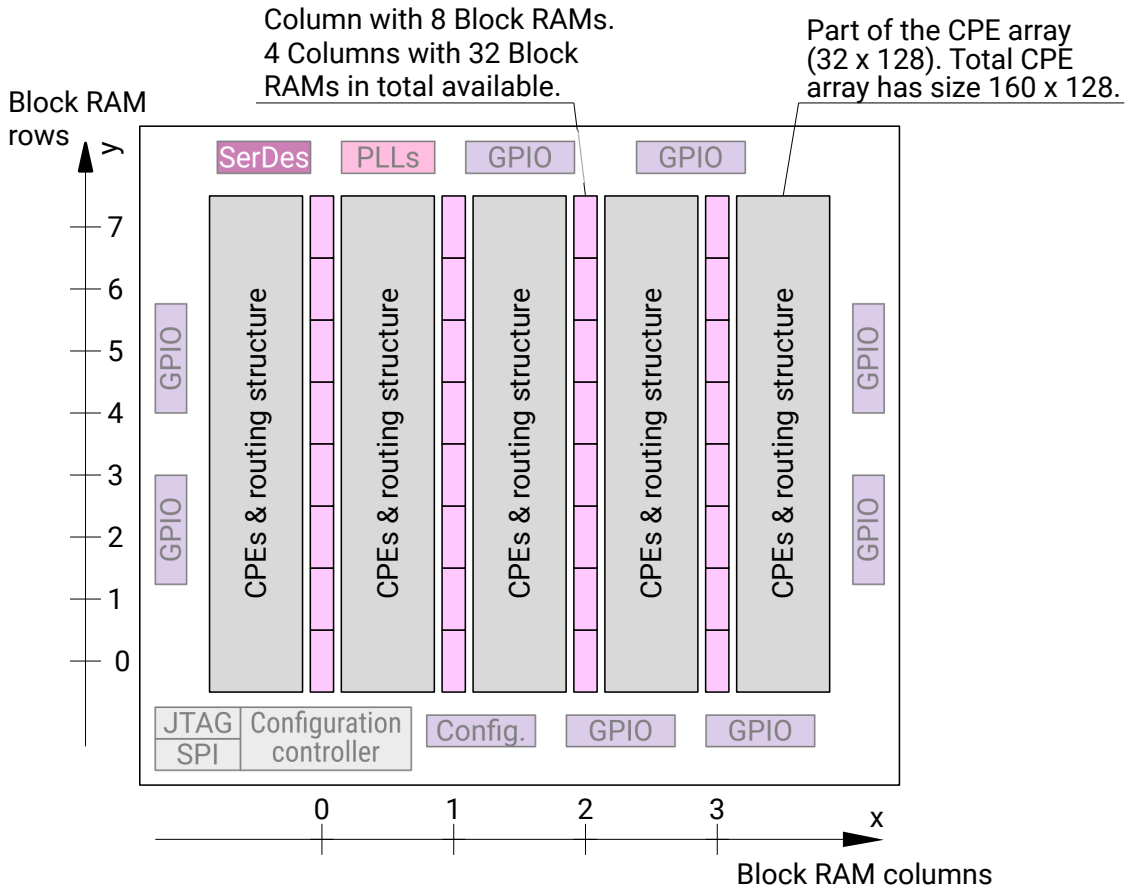


Figure 2.9: *Block RAM arrangement of CCGM1A1 (die view)*

Table 2.1: *Block RAM resources in GateMate™ Series*

Device	Total blocks	RAM columns	Blocks per column	Total memory bits
CCGM1A1	32	4	8	1,310,720
CCGM1A2	64	8	8	2,621,440

Each Block RAM cell can be configured as a single 40K or two independent 20K DPSRAM cells. Both allow usage of the memory in true dual port (TDP) or simple dual port (SDP)

mode.

The GateMate™ Block RAM features include:

- Data widths from 1 bit up to 40 bits in TDP mode or 80 bits in SDP mode.
- Bit-wide write enable allows a bit-wise writing of incoming data and can be used when for example interfacing with a microprocessor.
- Each port has optional output registers. When enabled, validity of the output data gets delayed by one clock cycle. This feature is particularly helpful when dealing with critical paths.
- Each port provides an error checking and correcting (ECC) module for data protection against unintended changes. Error correction is able to correct one bit error and detect two bit errors.
- A memory cascading feature connects adjacent RAM cells to form a larger array. This feature enables the instantiation of deeper or wider memories and a flexible forwarding of clocks, address and control signals.
- Two adjacent Block RAMs can be combined to one deeper 64K × 1 memory by extending the cascade feature to forward data and bitmask signals as well.
- Contents of the Block RAM cells can be initialized during configuration, e.g., to build read-only memory (ROM).
- All clock, enable and write enable signals can be inverted individually.
- Each Block RAM can be used as a first-in, first-out memory (FIFO) in asynchronous or synchronous mode with dedicated status signals for FIFO monitoring and reset.

Tables 2.2 and 2.3 show the available address and data bus width configurations as well as the ECC availability for both 20K and 40K configurations in SDP and TDP modes.

Table 2.2: *20K configurations*

Configuration	TDP	SDP	ECC
(RAM size per 20K block)			
16K × 1 bit	✓	✓	✗
8K × 2 bit	✓	✓	✗
4K × 5 bit	✓	✓	✗
2K × 10 bit	✓	✓	✗
1K × 20 bit	✓	✓	✗
512 × 40 bit	✗	✓	✓

Table 2.3: *40K configurations*

Configuration	TDP	SDP	ECC
32K × 1 bit	✓	✓	✗
16K × 2 bit	✓	✓	✗
8K × 5 bit	✓	✓	✗
4K × 10 bit	✓	✓	✗
2K × 20 bit	✓	✓	✗
1K × 40 bit	✓	✓	✓
512 × 80 bit	✗	✓	✓

2.4.2 Block RAM Architecture

A Block RAM cell has a maximum storage capacity of 40 Kbits and can be configured as a single 40K cell or two independent 20K cells. It contains two independent ports A and

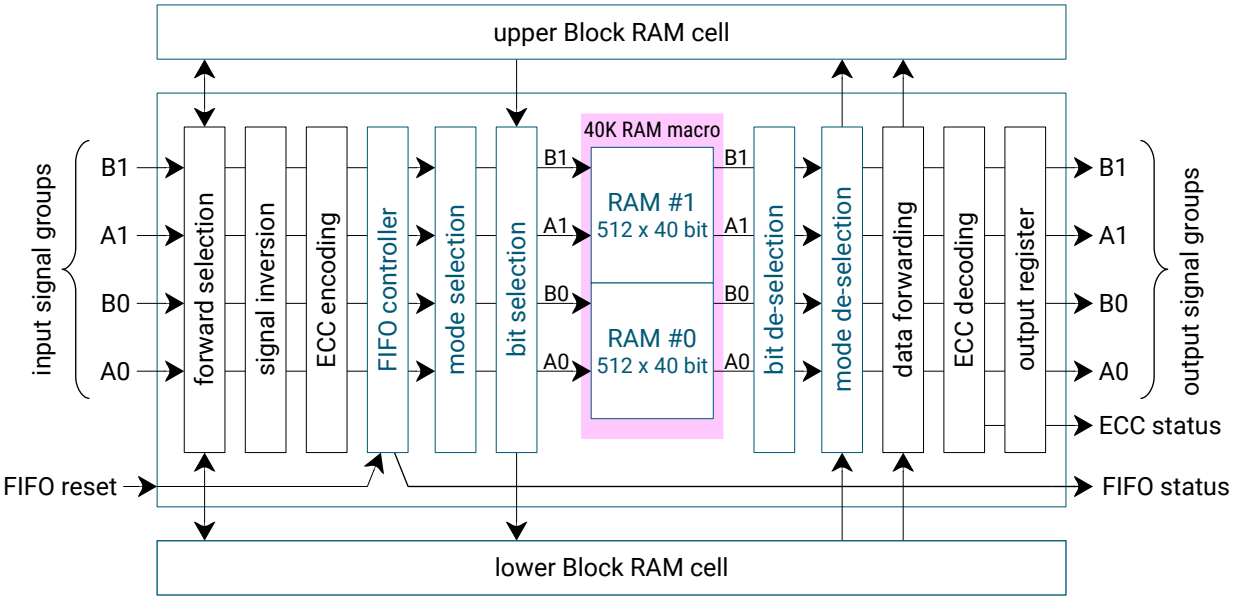


Figure 2.10: *High level block diagram of a block RAM cell*

B that are implemented in the form of the four input signal groups A0, B0, A1 and B1. Configured as two independent 20K cells, A0, B0, A1 and B1 represent ports A and B for the two cells. In the 40K configuration, A0+A1 and B0+B1 are combined to represent both ports A and B. Figure 2.10 shows the basic structure of a single Block RAM cell with its horizontal input and output connections from the CPE array and to the routing structure as well as the vertical interconnection to adjacent Block RAM cells.

The memory inputs can each be fed in from the CPE outputs RAM_O1 or RAM_O2. Each of the control and address signals CLK, EN, WE and ADDR can be selected from two input connections.

Table 2.4: *Input signal group*

Signals	Width	Description
{a0,a1,b0,b1}_clk{1,2}	1	Clock signal
{a0,a1,b0,b1}_en{1,2}	1	Enable signal
{a0,a1,b0,b1}_we{1,2}	1	Write enable signal
{a0,a1,b0,b1}_addr{1,2}	16	Address bus
{a0,a1,b0,b1}_di	20	Data bus
{a0,a1,b0,b1}_bm	20	Mask for bit-wise write enable

Data output signals and status output signals such as ECC and FIFO flags are connected via the CPE inputs RAM_I1 and RAM_I2 to the Switch Boxes (SBs) of the routing structure.

The entire functionality of the Block RAM is implemented over several layers around the RAM cores. The forward selection part allows a flexible switching between several in-

coming signals of which one can be selected and forwarded to the next layers. This corresponds to the clock, enable, write enable, address, data and bitmask signals. The forward selection determines whether signals (a) from an adjacent RAM cell, e.g., to allow the building of wider or deeper memories, or (b) from the CPE array via the input signal groups `A0`, `A1`, `B0` and `B1` are routed to the next layers.

In the signal inversion block the clock, enable and write enable signals are individually inverted depending on the configuration.

The Block RAM provides an internal ECC encoding and decoding layer to protect data against corruption. Depending on the configuration, 8 or 16 bits are required for storage of the parity bits. This feature is therefore only available for data widths of 40 or 80 bits as shown in Tables 2.2 and 2.3. Due to the storage of the parity bits, the actual net data width is 32 or 64 bits. Error correction is implemented using Hamming-code (39,32) with 7 parity bits. It can correct one bit error and detect two bit errors. The error status can be queried via the corresponding output signals. Section 2.4.7 describes the ECC features in more detail.

With the FIFO feature enabled, the internal write and read pointers will be forwarded to the memory macros. The address signals are therefore ignored. Port `B` is the push while port `A` is the pop side. The FIFO is configurable to be synchronous or asynchronous. Additional flags indicate the current status, such as empty, full, almost empty, almost full, read error, write error as well as the current FIFO read and write pointers. Moreover, the FIFO controller has a dedicated active low reset signal `FIFO_RST_N`. Section 2.4.9 describes the FIFO features in more detail.

The mode selection part forwards the incoming signals of ports `A0`, `A1`, `B0` and `B1` to the 8 ports of the memory hard macros according to the selected mode. Sections 2.4.3 and 2.4.4 describe the TDP and SDP modes in more detail.

Each port has an optional output register to improve design performance. If enabled, validity of the output data and ECC status flags get registered and thus delayed by one clock cycle.

2.4.3 TDP Mode

Block RAM in true dual port (TDP) mode supports simultaneous read and write operations and has two independent access ports. Figures 2.11 and 2.12 illustrate the TDP data flow in both 20K and 40K configurations.

Data can be read to or written from both ports at the same time. Read access to one port while writing to the other is also possible. Both ports can have separate clocks and input and output data widths may be different. Thus, the clocks can be synchronous or asynchronous. Both ports have access to the entire memory at any time.

There is no internal conflict handling when accessing the same address at the same time. Such simultaneous accesses can result in data uncertainty and should be avoided.

If split into two independent 20K cells, the maximum data width per port is 20 bits. In the 40K configuration, the maximum data width per port is 40 bits.

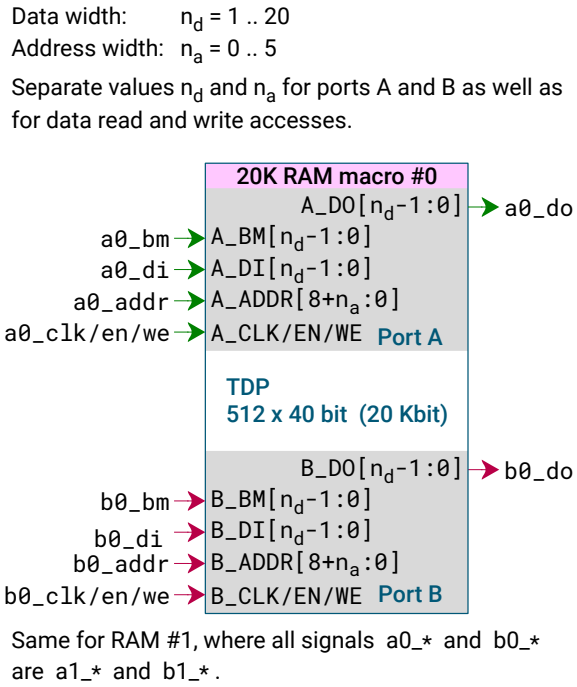


Figure 2.11: *Internal data flow of DPSRAM in TDP 20K mode*

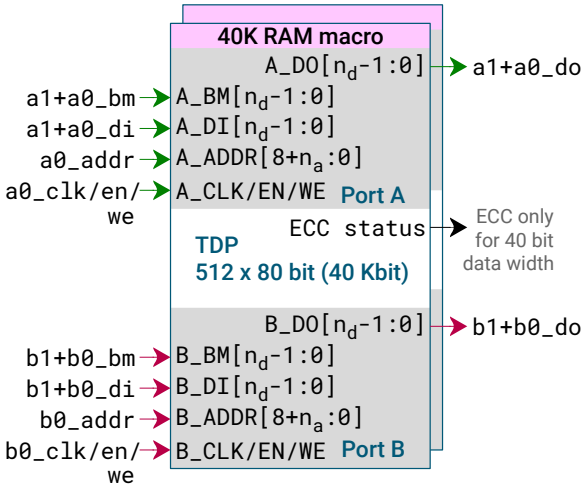


Figure 2.12: *Internal data flow of DPSRAM in TDP 40K mode*

Table 2.5 shows the data and address bus wiring in TDP 20K mode when using half Block RAM #0. Control signals are `a0_clk`, `a0_en` and `a0_we`. The wiring of `a0_di` also applies to the `a0_bm` signals. The wiring for the other half Block RAM #1 ports `A1` and `B1` is equivalent.

Table 2.5: *Pin wiring in TDP 20K mode*

Width	Depth	Address-in bus [1]	Data-in bus	Data-out bus
1	16,384	`a0_addr[15:7]` ∘ `a0_addr[5:1]`	`a0_di[0]`	`a0_do[0]`
2	8,192	`a0_addr[15:7]` ∘ `a0_addr[5:2]`	`a0_di[1:0]`	`a0_do[1:0]`
5	4,096	`a0_addr[15:7]` ∘ `a0_addr[5:3]`	`a0_di[4:0]`	`a0_do[4:0]`
10	2,048	`a0_addr[15:7]` ∘ `a0_addr[5:4]`	`a0_di[9:0]`	`a0_do[9:0]`
20	1,024	`a0_addr[15:7]` ∘ `a0_addr[5]`	`a0_di[19:0]`	`a0_do[19:0]`

[1] Symbol ∘ is concatenation.

Table 2.6 shows the data and address bus wiring in TDP 40K mode when using ports `A0` and `A1` to form port `A`. Control signals are `a0_clk`, `a0_en` and `a0_we`. Note that due to the port combining, the maximum data width per port is 40 bits. The wiring of `a0_di` also applies to the `a0_bm` signals. The wiring for port `B`, which is formed by `B0` and `B1`, is equivalent.

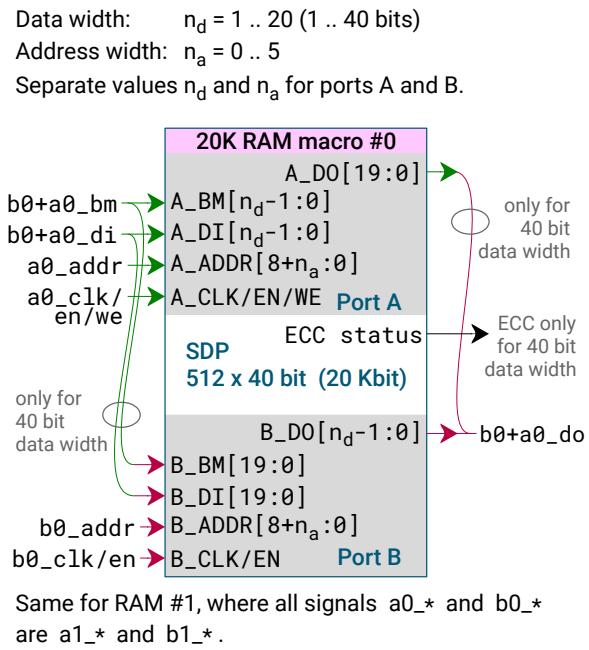


Figure 2.13: *Internal data flow of DPSRAM in SDP 20K mode*

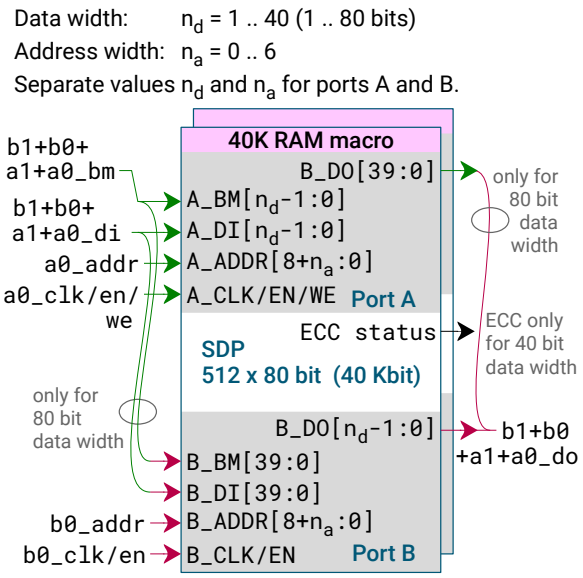


Figure 2.14: *Internal data flow of DPSRAM in SDP 40K mode*

2.4.4 SDP Mode

Block RAM in simple dual port (SDP) mode supports simultaneous read and write operations, but has a single output port for read data. By that, data widths can be doubled compared to the TDP mode.

Port A is always the write port, while port B is always the read port. Both write and read ports can have separate clocks. The write port A uses the enable signal {a0,a1}_en and write enable signal {a0,a1}_we for write access and the read port B uses only the enable signal {b0,b1}_en for read access.

Table 2.6: *Pin wiring in TDP 40K mode*

Width	Depth	Address-in bus	Data-in bus [1]	Data-out bus [1]
1	32,768	a0_addr[15:1]	a0_di[0]	a0_do[0]
2	16,384	a0_addr[15:2]	a0_di[1:0]	a0_do[1:0]
5	8,192	a0_addr[15:3]	a0_di[4:0]	a0_do[4:0]
10	4,096	a0_addr[15:4]	a0_di[9:0]	a0_do[9:0]
20	2,048	a0_addr[15:5]	a0_di[19:0]	a0_do[19:0]
40	1,024	a0_addr[15:6]	a1_di[19:0] ∘ a0_di[19:0]	a1_do[39:0] ∘ a0_do[39:0]

[1] Symbol ∘ is concatenation.

Table 2.7 shows the data and address bus wiring in SDP 20K mode. By combining the data buses on ports A0 and B0, the maximum bus width can be doubled compared to the TDP mode. Control signals for the write port A are a0_clk, a0_en and a0_we. Control signals for the read port B are b0_clk and b0_en. The wiring of {a0,b0}_di also applies to the {a0,b0}_bm signals. The wiring for ports A1 and B1 is equivalent.

Table 2.7: *Pin wiring in SDP 20K mode*

Width	Depth	Address-in bus	Data-in bus [1]	Data-out bus [1]
40	512	{b0,b0}_addr[15:7]	b0_di[19:0] ∘ a0_di[19:0]	b0_do[19:0] ∘ a0_do[19:0]

[1] Symbol ∘ is concatenation.

Table 2.8 shows the data and address bus wiring in SDP 40K mode. By combining the data buses on all ports A0, B0, A1 and B1, the maximum bus width can be doubled compared to the TDP mode. Control signals for the write port A are A0_CLK, A0_EN and A0_WE. Control signals for the read port B are b0_clk and b0_en. The wiring of {a0,a1,b0,b1}_di also applies to the {a0,a1,b0,b1}_bm signals.

Table 2.8: *Pin wiring in SDP 40K mode*

Width	Depth	Address-in bus	Data-in bus [1]	Data-out bus [1]
80	512	{b0,a0}_addr[15:7]	b1_di[19:0] ∘ b0_di[19:0] ∘ a1_di[19:0] ∘ a0_di[19:0]	b1_do[19:0] ∘ b0_do[19:0] ∘ a1_do[19:0] ∘ a0_do[19:0]

[1] Symbol ∘ is concatenation.

2.4.5 Expanding Data Widths and Cascade Mode

The Block RAM allows the coupling of adjacent Block RAMs to build larger or deeper memories by forwarding clock, enable, write enable and address signals. This feature facilitates the routing in the CPE array and the routing structure for the signals mentioned.

The cascade mode extends this feature by forwarding data and bitmask as well. It only allows a data width of 1 bit. Cascading is only possible in the lower direction, as shown in Figure 2.15 on page 38. By that, two adjacent 32K × 1 bit Block RAM cells can be combined to form a 64K × 1 bit memory. The actual data and bitmask inputs and outputs of the cascade are the data and bitmask inputs and outputs of the upper RAM cell. Prerequisite is that both RAM cells involved are configured in 32K × 1 bit TDP mode. Bit 0 of the address bus selects the upper or lower Block RAM cell for the write or read operation.

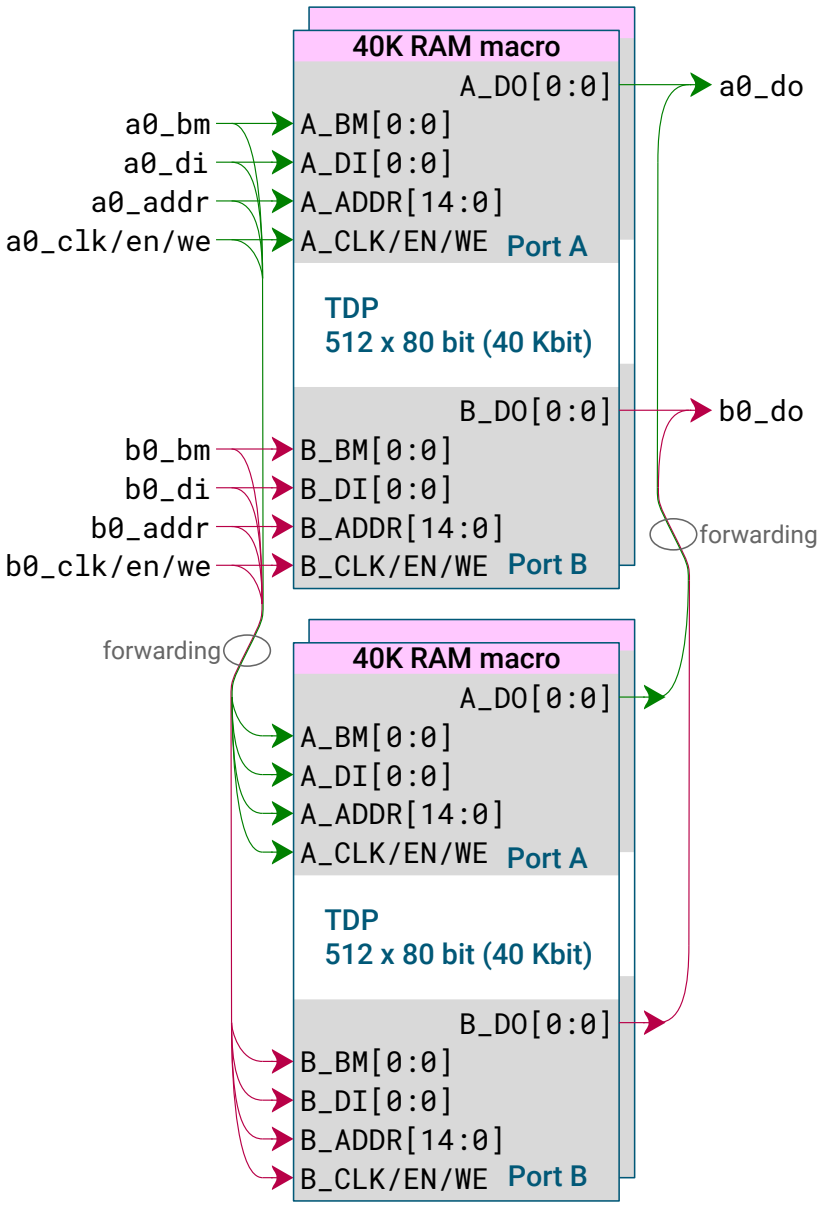


Figure 2.15: *Structure of two adjacent DPSRAMs forming one 64K × 1 bit memory via cascade feature*

2.4.6 Memory Mapping and Content Initialization

Each port can access a certain memory area depending on its configuration. The addressing scheme depends on the configuration mode. Words of 5, 10, 20, 40 and 80 [2] bits are distributed equally over all four static random-access memory (SRAM) blocks whereas 1 and 2 bit wide words are arranged so that every 5th bit is not accessible. This leads to a logical mapping of the memory bits as shown in Figure 2.16 on page 39. The scheme shown here can be applied to both SDP and TDP modes in 20K as well as in 40K configurations.

[2] Data width of 80 bits only supported in 40K SDP mode (see Tables 2.2 and 2.3).

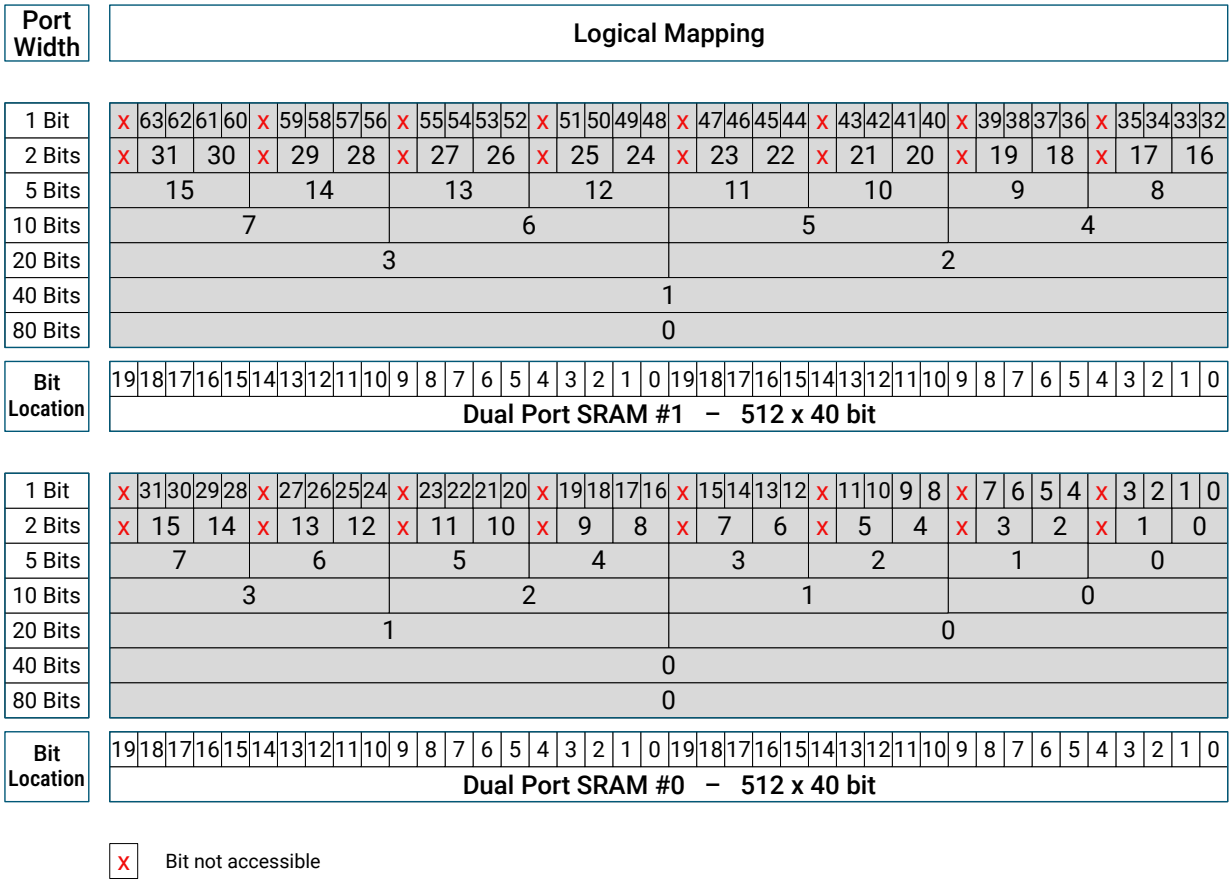


Figure 2.16: *Logical data mapping of memory at physical address 0*

2.4.7 ECC Encoding / Decoding

The Block RAM cell provides an internal ECC encoding and decoding layer to protect data against corruption. If configured, parity bits are generated and checked automatically. Using this feature, the user cannot utilize the all 40 or 80 data bits but must restrict to 32 or 64 bits since the remaining 8 or 16 bits are required for storage of parity bits. This feature is only available for the following configurations:

- TDP 40K, configured to 40 bits of which user can only use 32 bits
- SDP 40K, configured to 80 bits of which user can only use 64 bits
- SDP 20K, configured to 40 bits of which user can only use 32 bits

In any case the bitmask is ignored, all 32 or 64 bits plus parity are written to memory. For error correction the Hamming-code (39,32) with 7 parity bits is used. It can correct one bit error and detect two bit errors. If one error was detected the ECC single error flag signal will be logic `true`. If two errors were detected the ECC double error flag will be logic `true`. There are two dedicated status signals `{A,B}_ECC_{1B,2B}_ERR` for single and double error flags each, which are utilized in the following way:

- TDP 40K, data width 40 bits: status signal `A_ECC_{1B,2B}_ERR` indicate errors at port `A` while `B_ECC_{1B,2B}_ERR` indicate errors at port `B`

- SDP 40K, data width 80 bits: status signal `A_ECC_{1B,2B}_ERR` indicate errors while `B_ECC_{1B,2B}_ERR` are don't care
- SDP 20K, data width 40 bits: status signal `A_ECC_{1B,2B}_ERR` indicate errors at half Block RAM #0 while `B_ECC_{1B,2B}_ERR` indicate errors at half Block RAM #1

2.4.8 RAM Access Modes and Enable

A block RAM cell has three signals to control read and write access. `{A|B}_EN` is a global enable signal. It is required to be active during any read or write access. `{A|B}_WE` is a global write enable signal that operates in association with the bit-wise write enable vector `{A|B}_BM`.

Data is only written to the RAM if `{A|B}_EN`, `{A|B}_WE` and the corresponding bitmask `{A|B}_BM` are active. Reading of data depends on the selected access mode and the values of `{A|B}_WE` and `{A|B}_BM`. The address modes are listed in Table 2.9 and 2.10. They show all combinations of `{A|B}_EN`, `{A|B}_WE` and `{A|B}_BM` in both modes `NO CHANGE` and `WRITE THROUGH` as well as the resulting actions.

Table 2.9: *Access combinations in NO CHANGE mode with {A|B}_EN = 1 (SDP and TDP)*

{A|B}_WE [1]	{A|B}_BM[i]	Action on memory and {A|B}_DO
0	0	Single read, no update on `mem[addr][i]`
1	0	Last read, no update on `mem[addr][i]`
0	1	Single read, no update on `mem[addr][i]`
1	1	Last read, with update on `mem[addr][i]`

[1] Only `A_WE` in SDP mode.

Table 2.10: *Access combinations in WRITE THROUGH mode with {A|B}_EN = 1 (TDP only)*

{A|B}_WE	{A|B}_BM[i]	Action on memory and {A|B}_DO
0	0	Single read, no update on `mem[addr][i]`
1	0	Single read, no update on `mem[addr][i]`
0	1	Single read, no update on `mem[addr][i]`
1	1	Write through, with update on `mem[addr][i]`

A *single read* access without any write operation occurs always if the enable signal is active (`{A|B}_EN` = 1) and all write enable signals are inactive (`{A|B}_BM` = 0, `{A|B}_WE` = 0). A single read access is supported in both modes SDP and TDP as shown in Figure 2.17.

All RAM cells can be initialized during configuration and used as read-only memory (ROM). To use the ROM mode the write enable has to be set to zero. The memory mapping for initialization is described in Section 2.4.6.

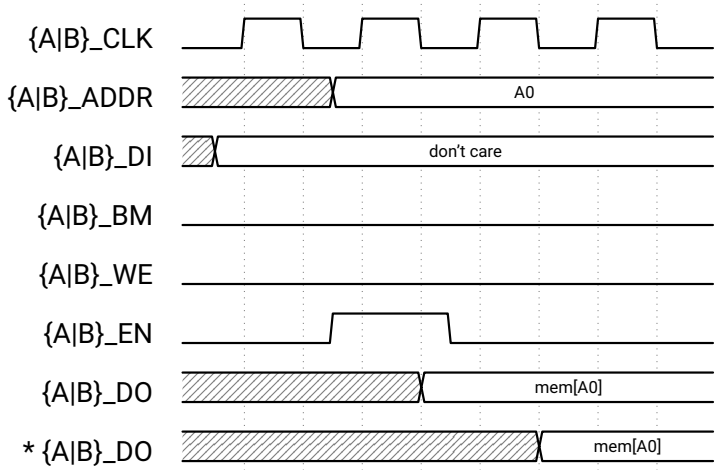


Figure 2.17: *Timing diagram of a single read access with optional (*) output register*

The NO CHANGE mode is a plain write access with active global enable ({A|B}_EN = 1) and write enable signals ({A|B}_BM >= 1 or {A|B}_WE = 1). The output remains the last read data, i.e. after a single read, and is not affected by a write access. A NO CHANGE access is supported in both modes SDP and TDP as shown in Figure 2.18.

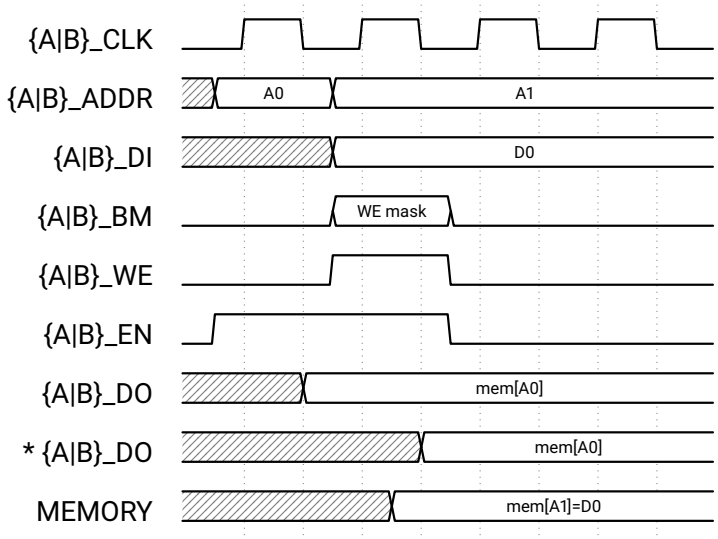


Figure 2.18: *Timing diagram of a NO CHANGE access with optional (*) output register*

A read first access can be carried out in two clock cycles by first reading a word and then performing a write access in the following cycle.

The timing diagram 2.19 illustrates the WRITE THROUGH access, which is a simultaneous write and read access with active global enable ({A|B}_EN = 1) and write enable signals ({A|B}_BM >= 1 or {A|B}_WE = 1). New data is written into the memory and simultaneously propagated to the outputs, also known as a transparent write. WRITE THROUGH is only supported in TDP mode.

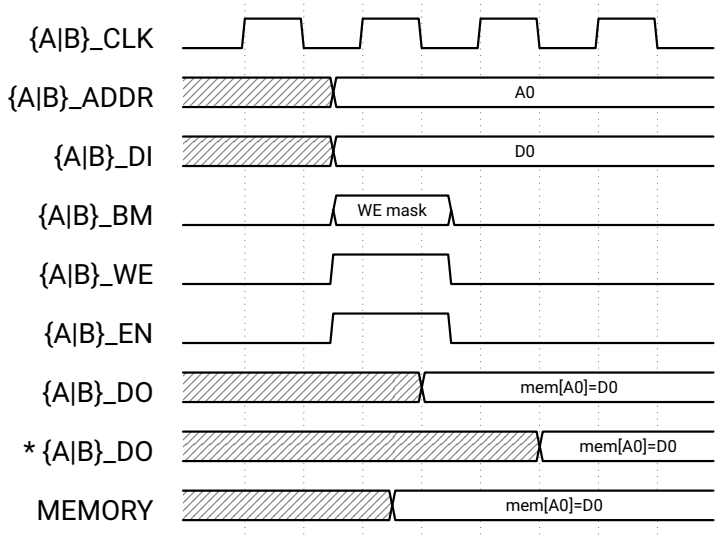


Figure 2.19: *Timing diagram of a `WRITE THROUGH` access with optional (*) output register*

2.4.9 FIFO Application

Each GateMate™ block RAM cell has an integrated synchronous and asynchronous FIFO controller, allowing usage of a block RAM cell as FIFO memory in the 40K configurations only.

Port `B` is the write / push port of the FIFO and port `A` is the read / pop port. In case of synchronous FIFO, `A_CLK` is used as clock for both write / push and read / pop.

Since the FIFO mode is an extension to the TDP / SDP 40K mode, it supports the same bitwidth configurations as shown in Tables 2.6 (page 36) and 2.8. Widths of the input and output buses must be equal.

- TDP 40K, with arbitrary but equal input and output bit width
- SDP 40K, with fixed 80 bit input and output bit width

In FIFO configuration, the incoming address signals are ignored. Instead, internal read and write pointers with correct alignment according to bitwidth are forwarded to the SRAM macros. Furthermore, there exist additional output signals which are solely used for FIFO monitoring and are described in Table 2.11. The `F_ALMOST_FULL_FLAG` and `F_ALMOST_EMPTY_FLAG` give an early warning when the FIFO is approaching its limits. Its offset can be configured using 15 bit registers during configuration or it is set dynamically using the inputs `F_ALMOST_FULL_OFFSET` and `F_ALMOST_EMPTY_OFFSET`.

Please note, that the utilization of dynamic offset inputs is exclusively accessible within the TDP configuration. In SDP mode, the static offset configuration remains the sole viable option.

Moreover, the FIFO controller has a dedicated active low reset input signal `F_RST_N` which is synchronized into clock domain internally.

Table 2.11: *FIFO status flags*

Flag	Width	Description
F_FULL	1	All entries in the FIFO are filled, is set on the rising edge of the write clock (asynchronous)
F_EMPTY	1	The FIFO is empty, is set on the rising edge of the read clock (asynchronous)
F_ALMOST_FULL	1	Almost all entries in the FIFO are filled, is set on the rising edge of the write clock (asynchronous)
F_ALMOST_EMPTY	1	Almost all entries in FIFO have been read, is set on the rising edge of the read clock (asynchronous)
F_READ_ADDRESS	16	Current FIFO read pointer
F_WRITE_ADDRESS	16	Current FIFO write pointer
F_RD_ERR	1	Is set if FIFO is empty and a read access takes place
F_WR_ERR	1	Is set if FIFO is full and data is pushed, new data will be lost

Table 2.12 illustrates the valid FIFO data concatenations for the variable bitwidth data TDP mode. B_EN and B_WE are the write / push enable and A_EN is the read / pop enable signal.

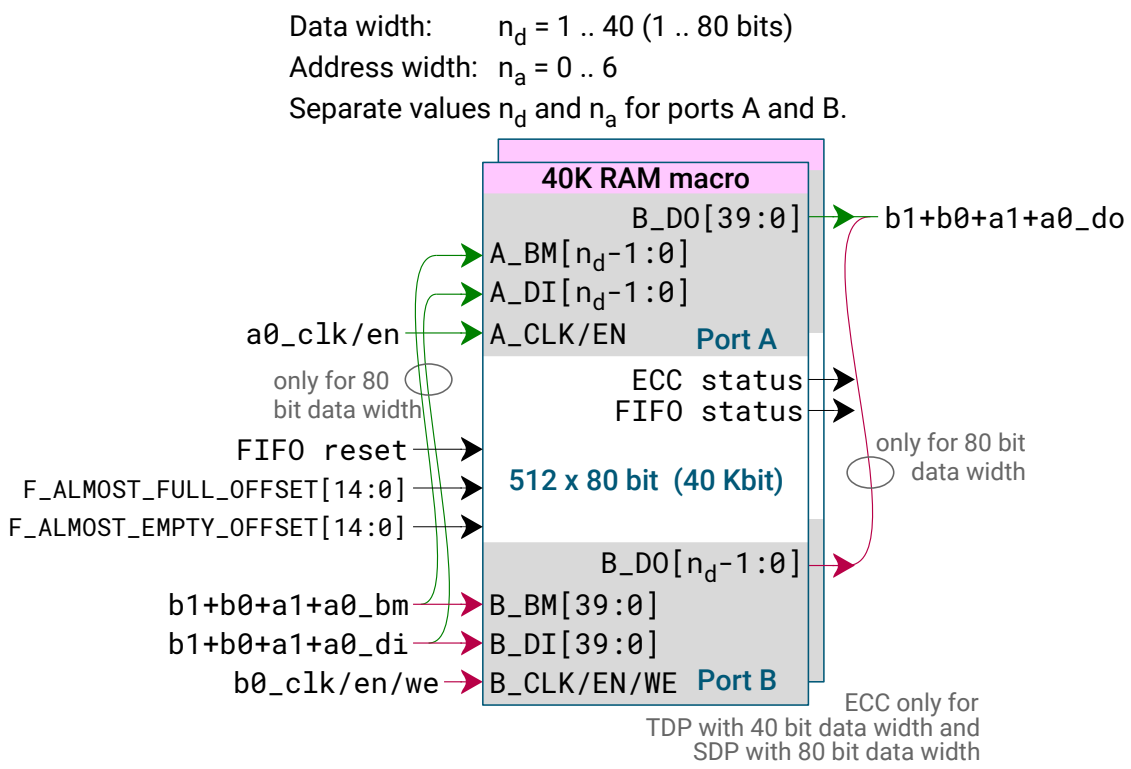


Figure 2.20: *Internal data flow of DPSRAM in FIFO mode*

Table 2.12: *FIFO 40 bit data concatenations*

Function	Width	Concatenation
Push data	40	`B_DI[39:0]`
Push bitmask	40	`B_BM[39:0]`
Pop data	40	`A_DO[39:0]`

Table 2.13 illustrates the valid FIFO data concatenations for the 80 bit data SDP mode. `B_EN` and `B_WE` are the write / push enable and `A_EN` is the read / pop enable signal.

Table 2.13: *FIFO 80 bit data concatenations*

Function	Width	Concatenation
Push data	80	`B_DI[39:0]` ∘ `A_DI[39:0]`
Push bitmask	80	`B_BM[39:0]` ∘ `A_BM[39:0]`
Pop data	80	`B_DO[39:0]` ∘ `A_DO[39:0]`

[1] Symbol ∘ is concatenation.

Synchronous FIFO Access

Write / push and read / pop pointers are both registered with the rising clock edge of `A_CLK`. During the write / push operation, the data word available at `{A,B}_DI` / `{A,B}_BM` is written into the FIFO whenever the `B_EN` and `B_WE` signals are active one setup time before the rising clock edge of `A_CLK`. The write / push operation presents the data word at `{A0,A1}_DO` whenever the `A_EN` signal is active one setup time the rising clock edge of `A_CLK`.

The signals empty, full, almost empty, almost full, read and write error are combinatorial computed out of read and write pointer. The error flags are not sticky.

The timing diagram in Figure 2.21 illustrates the writing to an empty synchronous FIFO.

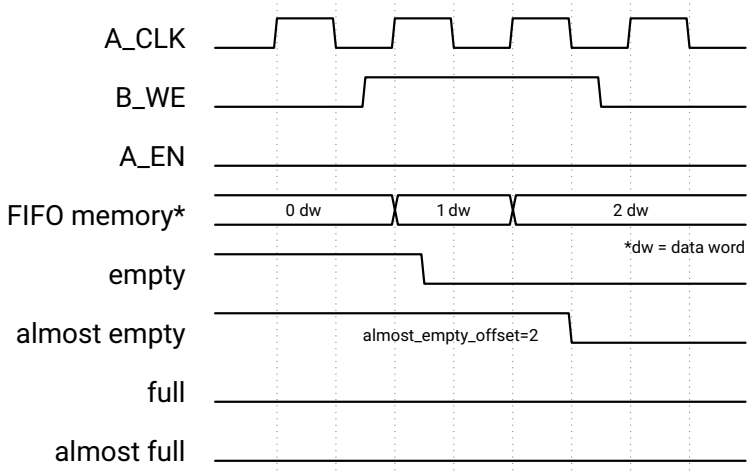


Figure 2.21: *Writing to an empty synchronous FIFO*

The timing diagram in Figure 2.22 illustrates the writing to an almost full synchronous FIFO.

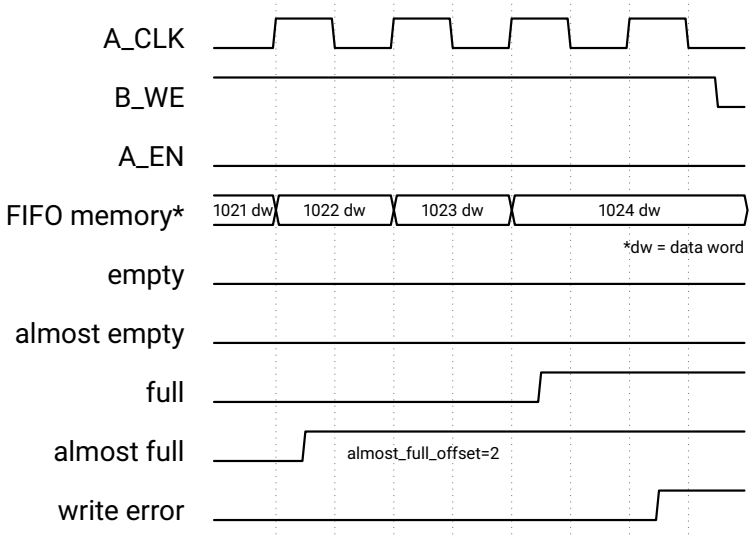


Figure 2.22: *Writing to an almost full synchronous FIFO*

The timing diagram in Figure 2.23 illustrates the reading from a full synchronous FIFO.

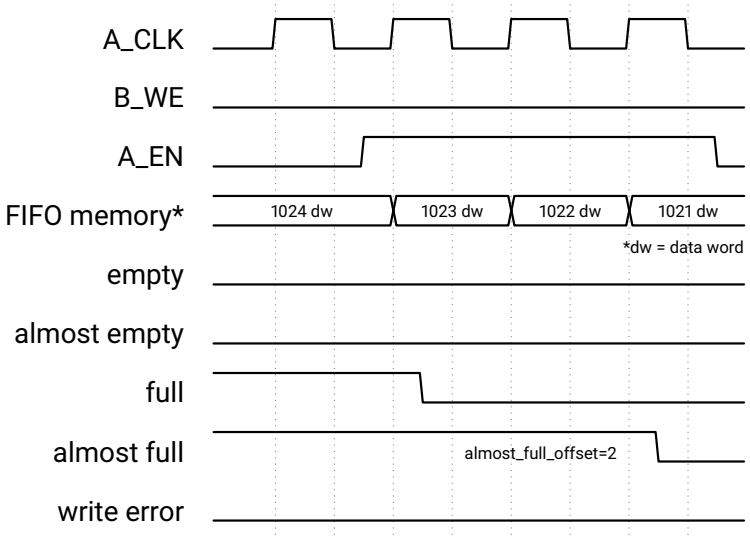


Figure 2.23: *Reading from a full synchronous FIFO*

The timing diagram in Figure 2.24 illustrates the reading from an almost empty synchronous FIFO.

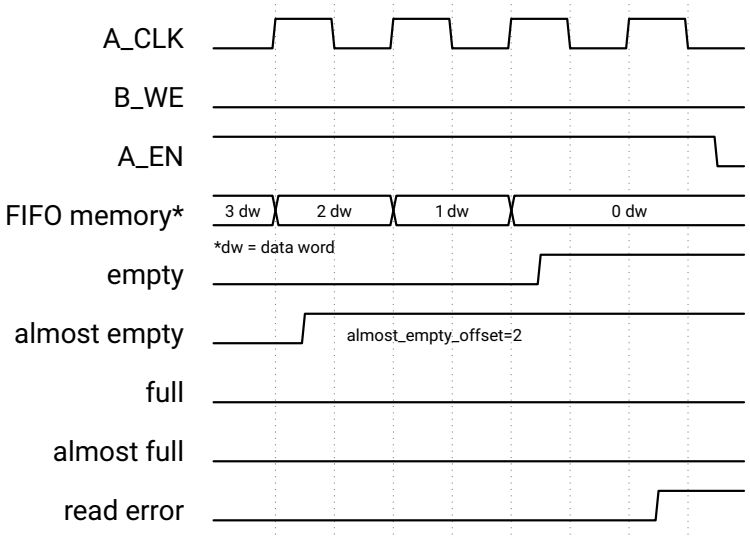


Figure 2.24: *Reading from an almost empty synchronous FIFO*

Asynchronous FIFO Access

During the write / push operation, the data word available at {A,B}_DI / {A,B}_BM is written into the FIFO whenever the B_EN and B_WE signals are active one setup time before the rising clock edge of B_CLK. The write / push operation presents the data word at {A,B}_DO whenever the A_EN signal is active one setup time the rising clock edge of A_CLK. The read / pop pointer is registered with the read / pop clock A_CLK, the write / push pointer is registered with the write / push clock B_CLK.

For full, empty, almost full, almost empty, read and write error signal generation the read pointer is synchronized into the write clock domain via Gray encoding and 2-stage synchronization, the write pointer is synchronized into the read clock domain via Gray encoding and 2-stage synchronization.

The empty, almost empty and read error signals are combinatorial computed out of the read / pop pointer and the synchronized write / push pointer. By that, the empty, almost empty and read error signals will be always cycle accurate with the read clock without any delay and a read from an empty FIFO can always be prevented.

In Figures 2.25 to 2.28, these signals are highlighted in red relate to the read side of the FIFO since their computation bases on flip-flops driven by the read / pop clock. The full, almost full and write error signals are combinatorial computed out of the write / push pointer and the synchronized read / pop pointer. By that, the full, almost full and write error signals will be always cycle accurate with the write clock without any delay and a write to a full FIFO can always be prevented.

Moreover, the signals highlighted in green are related to the write side of the FIFO since their computation bases on flip-flops driven by the write / push clock.

The timing diagram in Figure 2.25 illustrates the writing to an empty asynchronous FIFO.

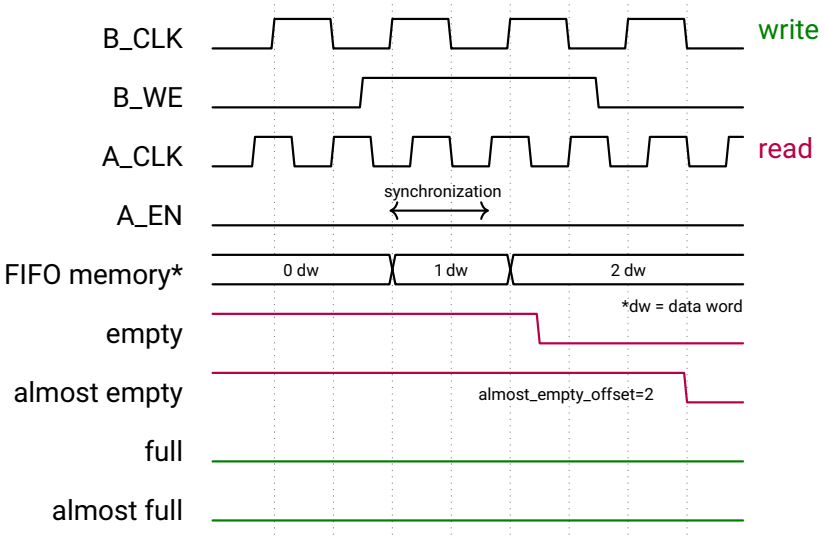


Figure 2.25: *Writing to an empty asynchronous FIFO*

The timing diagram in Figure 2.26 illustrates the writing to an almost full asynchronous FIFO.

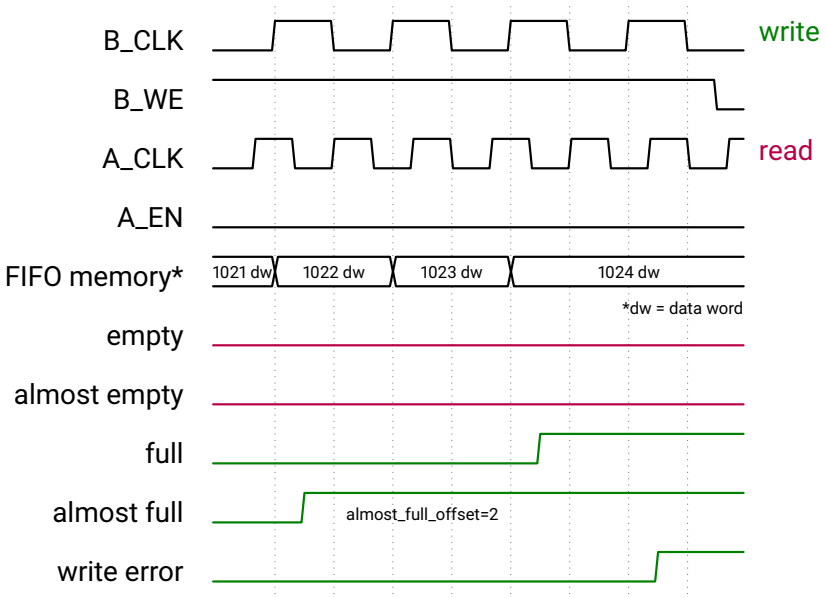


Figure 2.26: *Writing to an almost full asynchronous FIFO*

The timing diagram in Figure 2.27 illustrates the reading from a full asynchronous FIFO.

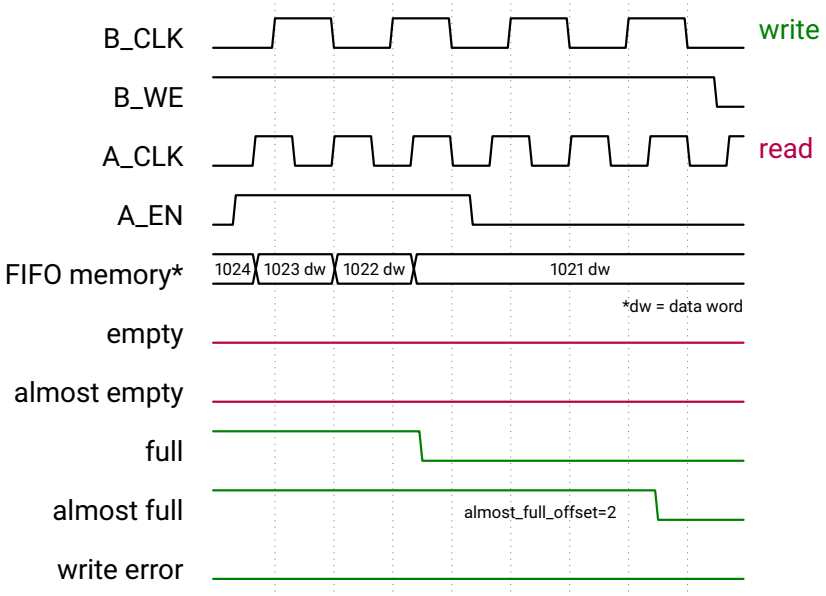


Figure 2.27: *Reading from a full asynchronous FIFO*

The timing diagram in Figure 2.28 illustrates the reading from a full asynchronous FIFO.

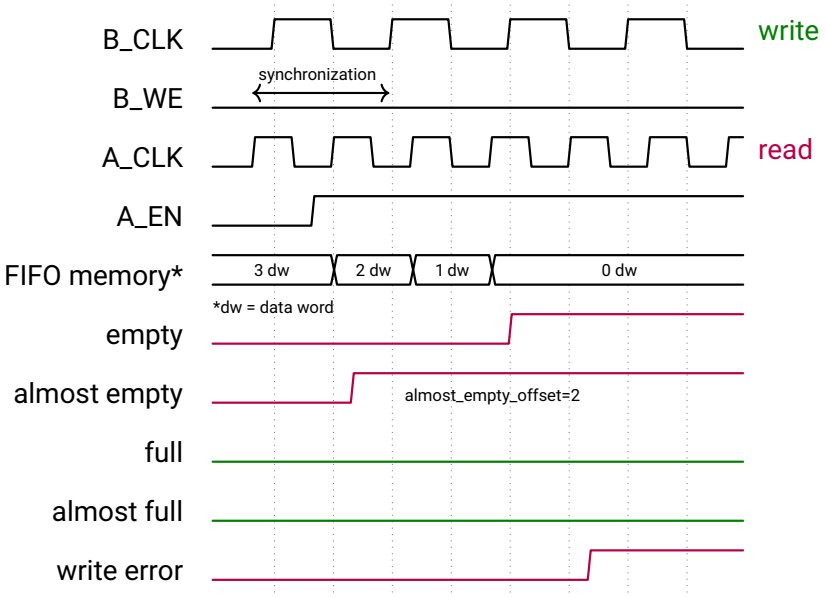


Figure 2.28: *Reading from an almost empty asynchronous FIFO*

2.5 Serializer / deserializer (SerDes)

2.5.1 SerDes Architecture Overview

This section describes the structure, features and functions of the build-in serializer / deserializer (SerDes) for high speed serial data transfers.

The GateMate™ SerDes features include:

- Line rate up to 5 Gbit/s
- Built-in low-jitter all-digital phase-locked loop (ADPLL)
- Physical coding sublayer (PCS)
  - Configurable 16 / 20-, 32 / 40- or 64 / 80-bit transmitter and receiver data pathes
  - 8B / 10B encoding and decoding
  - Comma detection and byte alignment
  - Transmitter Clock and Data Recovery (CDR)
  - Multiple pseudo-random bit stream (PRBS) generators and checkers
  - Phase adjust FIFO for clock correction (elastic buffer)
  - Polarity control
- Physical media attachment (PMA)
  - 3-tap decision feedback equalizer (DFE)
  - Transmitter pre- and post-emphasis
  - Configurable transmitter driver

The core of the SerDes is composed of an ADPLL, a transmitter (TX) and a receiver (RX). It is configured via an integrated register file, which can be accessed both from the FPGA

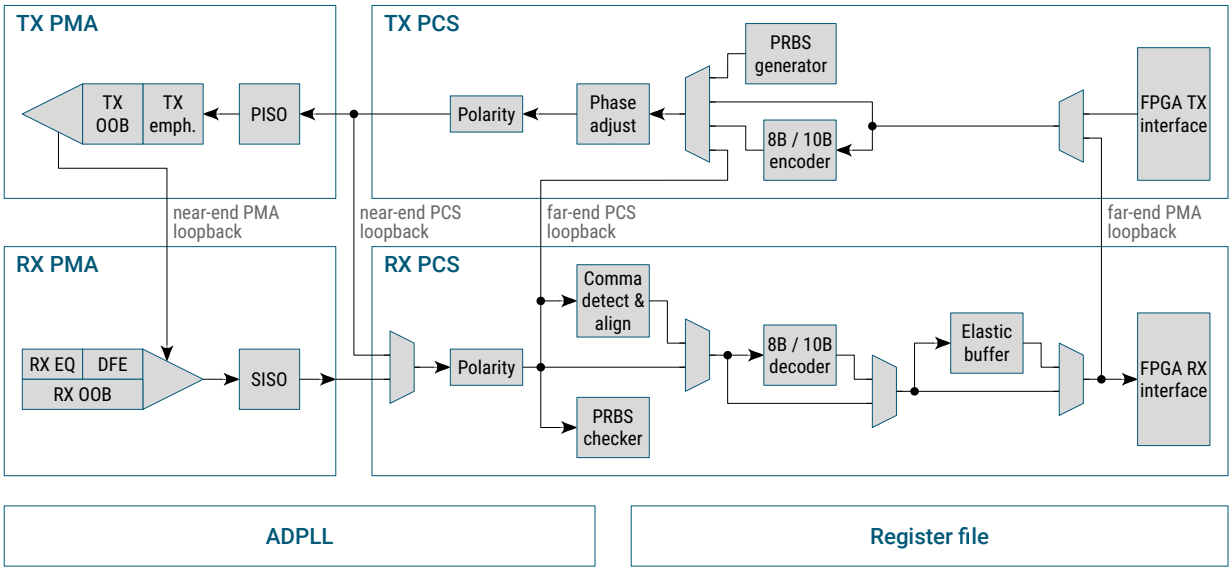


Figure 2.29: *Simplified SerDes blockdiagram*

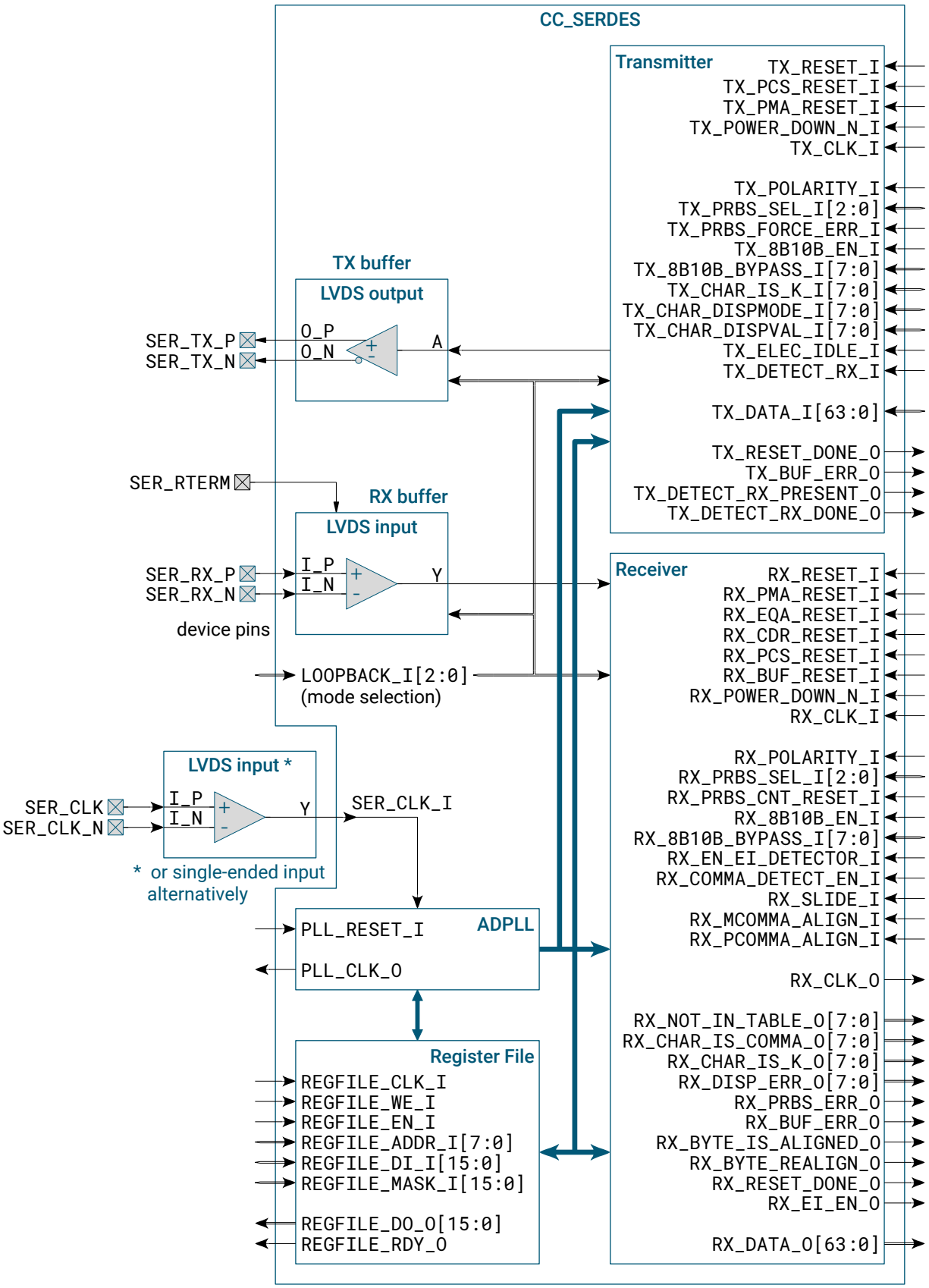


Figure 2.30: *CC_SERDES primitive blockdiagram*

fabric and the configuration controller. The ADPLL of the SerDes generates the bit rate clock for the serial transmission from an external reference clock. Additionally, it generates the clock for the operation of the receiver and transmitter data paths.

The transmitter and receiver data pathes of the SerDes consists of the components shown in Figure 2.29, most of which can be controlled via the FPGA interface.

Within the transmitter data path, the SerDes' PCS features an 8b / 10b encoder, a pattern generator for generating pseudo-random bit stream (PRBS) and a function block for inverting the polarity of the transmit data. The phase adjust FIFO function block, which is also referred to as the transmitter buffer, is used to align the phase shift between the clock domains used in the PCS and PMA blocks. The phase adjustment is mandatory for the transfer of the parallel data to the serial transmission, which is represented by the function block parallel-in serial-out (PISO). The actual transmitter or ransmit driver at line level allows feed-forward equalization for the transmission and the analog operating parameters of the transmit driver are configurable via the register file of the SerDes.

On the receiver data path, the SerDes' PCS features - analogous to the transmitter data path - a function block for inverting the polarity of the received data, a control unit for checking the PRBS and a 8b / 10b decoder. The receiver data path also features a comma detection and symbol alignment unit and an elastic buffer to compensate for clock tolerances. Moreover, the receiver PCS has a Clock and Data Recovery (CDR) unit. The receiver's analog front end (PMA) features a linear equalizer (receiver EQ) followed by a DFE.

For other use cases, the SerDes has multiple direct interconnections between the transmitter and receiver data paths to provide loopback of data at certain levels of the data path.

The SerDes primitive is shown in Figure 2.30 on page 51. Please refer to

UG1001 – GateMate^{TM} FPGA Primitives Library

for more information how to use the primitive.

2.5.2 SerDes ADPLL

The all-digital phase-locked loop (ADPLL) is a low-jitter, high-performance clocking module designed to deliver precise timing control within the SerDes block of the FPGA. It utilizes a dedicated reference clock input (pins SER_CLK and SER_CLK_N), configurable for single-ended or LVDS operation, and supports a reference clock frequency range of 100 MHz to 125 MHz.

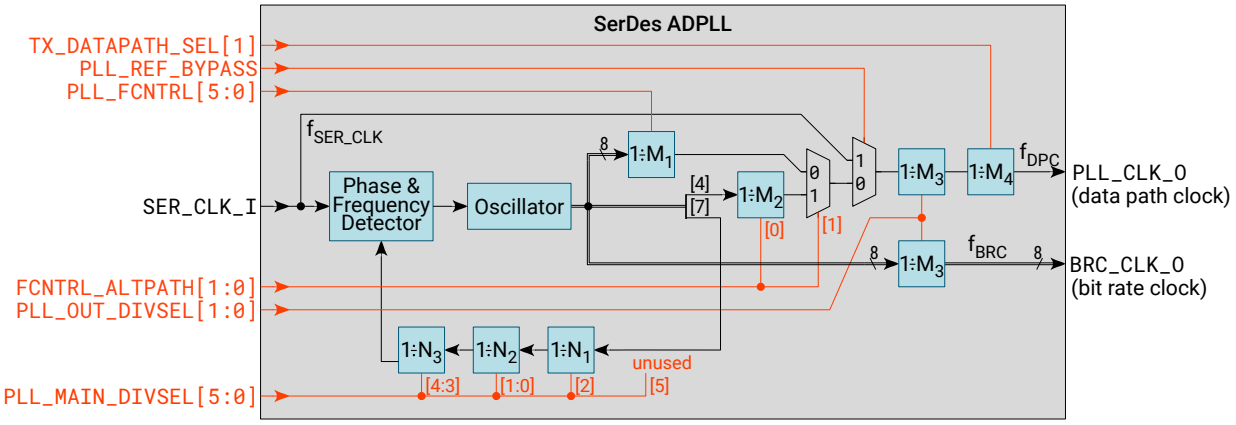


Figure 2.31: *Simplified SerDes ADPLL overview*

Table 2.14: *CC_SERDES PLL port description*

Port	Width	Direction	Description
PLL_RESET_I	1	Input	Asynchronous ADPLL reset
SER_CLK_I	1	Input	SerDes clock input
PLL_CLK_O	1	Output	Data path clock output

Using the provided reference clock, the ADPLL generates the required bit rate clock (BRC) for serial data transmission. Additionally, it generates the necessary data path clock (DPC) for the operation of the receiver and transmitter data paths at the PCS level. Table 2.14 shows the related ADPLL interface ports.

The core clock signal is provided by the ADPLL through the output port PLL_CLK_O. This clock signal must be connected within the FPGA logic to the transmitter and receiver interface ports TX_CLK_I and RX_CLK_I, respectively. For further details, refer to the data path related ports in Sections 2.5.5.1 and 2.5.6.1.

Alternatively, for the receiver data path, the clock signal recovered from the received data can be used. This clock signal is output via the port RX_CLK_O. For further details, refer to the Clock and Data Recovery (CDR) in Table 2.47 on page 81.

Table 2.15: *ADPLL clock divider settings in the register file*

Register File

Name	Width	Mode	Default	Description
PLL_MAIN_DIVSEL	6	r/w	27	[1:0] = 00 : $N_2 = 3$ [1:0] = 01 : $N_2 = 2$ [1:0] = 10 : $N_2 = 4$ [1:0] = 11 : $N_2 = 5$ (default) [2] = 0 : $N_1 = 1$ (default) [2] = 1 : $N_1 = 2$ [4:3] = 00 : $N_3 = 3$ [4:3] = 01 : n.a. [4:3] = 10 : $N_3 = 4$ [4:3] = 11 : $N_3 = 5$ (default) [5] = 0 : unused, do not change
PLL_OUT_DIVSEL	2	r/w	0	00: $M_3 = 1$ 01: $M_3 = 2$ 10: n.a. 11: $M_3 = 4$
PLL_FCNTRL	6	r/w	58	Frequency divider M_1 selection for SerDes transmitter clock output f_{DPC} to FPGA core (see Table 2.17 for M_1 values)

Table 2.16: *Alternative clock path settings with FCNTRL_ALTPATH*

PLL_FCNTRL	FCNTRL_ALTPATH	Multi-plexer	M_2 value	Description
0b 00 0000	0b 10	1	1	Alternative clock path through M_2
0b 00 0001	0b 11	1	2	Alternative clock path through M_2
else	0b 01	0	X	Normal clock path through M_1

Table 2.17: *ADPLL M_1 frequency divider settings*

PLL_FCNTRL	M_1	PLL_FCNTRL	M_1
0	*1	32	9
1	*1	33	10.5
2	3	34	9
3	3.5	35	10.5
4	3.5	36	10.5
5	3.75	37	11.25
6	4	38	12
7	4	39	12
8	4.5	40	13.5
9	5.25	41	15.75
10	5.25	42	15
11	5.5	43	16.5
12	5.5	44	16.5
13	5.75	45	17.25
14	6	46	18
15	6	47	18
16	6	48	12
17	7	49	14
18	6	50	12
19	7	51	14
20	7	52	14
21	7.5	53	15.75
22	8	54	16.5
23	8	55	16.5
24	9	56	18
25	10.5	57	21
26	10	58	20
27	11	59	22
28	11	60	22
29	11.5	61	23
30	12	62	24
31	12	63	24

*1 These values are used to determine the parameter FCNTRL_ALTPATH, see Table 2.16

The clock frequencies for both the bit rate clock (BRC) and the data path clock (DPC) are determined based on the selected reference clock frequency f_{SER_CLK} by

$$f_{BRC} = f_{SER\_CLK} \cdot \frac{N_1 \cdot N_2 \cdot N_3}{M_3} \tag{2.1}$$

and

$$f_{DPC} = f_{SER\_CLK} \cdot \frac{N_1 \cdot N_2 \cdot N_3}{M_3} \cdot \frac{1}{M_1 \cdot M_4} = \frac{f_{BRC}}{M_1 \cdot M_4} \tag{2.2}$$

with the parameters $N_1 \,..\, N_3$ and M_3 described in Table 2.15. The parameters have a predefined value range and can be configured via the corresponding register fields of the SerDes. The divider value M_1 can be set via the register field PLL_FCNTRL. Table 2.17 shows all available settings. Please note, that PLL_FCNTRL $= 0$ and PLL_FCNTRL $= 1$ do not set M_1 value but have a different usage for the alternative clock path. See Table 2.16 for the FCNTRL_ALTPATH setup.

If the 64 bit data path is used, TX_DATAPATH_SEL[1] $= 1$ is set, whereby an additional divider $M_4 = 2$ is inserted into the clock path. For all other data paths, no additional divider is required, which means TX_DATAPATH_SEL[1] $= 0$, and the output of M_3 is routed directly to the PLL output PLL_CLK_0.

2.5.2.1 ADPLL Clock Configuration Example

The ADPLL generates several phase-shifted clock signals with regard to the bit rate clock. These are used by the SerDes to double the transmission rate for this bit rate clock. For example, a transmission rate of 2.5 Gbit/s will require a frequency of 1.25 GHz, which can be realized with equation 2.1 and the following settings when using an exernal 100 MHz reference clock:

$$f_{BRC} = 100\,\text{MHz} \cdot \frac{N_1 \cdot N_2 \cdot N_3}{M_3} = 100\,\text{MHz} \cdot \frac{1 \cdot 5 \cdot 5}{2} = 1.25\,\text{GHz} \tag{2.3}$$

Assuming a transfer rate of $2f_{BRC} = 2.5\,\text{Gbit/s}$ and a 64 bit wide data path, the required frequency for the core clock is

$$f_{DPC} = \frac{2.5\,\text{Gbit/s}}{80\,\text{Bit}} = 31.25\,\text{MHz}\ . \tag{2.4}$$

Please note, that the 64 bits are transmitted as 80 bits according to 8B / 10B coding.

The frequency can be set with equation 2.2 using the dividers M_1, taking the $M_4 = 2$ divider into account:

$$f_{DPC} = \frac{f_{BRC}}{M_1 \cdot M_4} = \frac{1.25\,\text{GHz}}{20 \cdot 2} = 31.25\,\text{MHz} \tag{2.5}$$

2.5.2.2 ADPLL Advanced Configuration

Table 2.18: *ADPLL advanced configuration in the register file*

Register File

Name	Width	Mode	Default	Description
PLL_CI	5	r / w	3	Integral coefficient of loop filter
PLL_CP	10	r / w	12	Proportional coefficient of loop filter
PLL_OPEN_LOOP	1	r / w	0	ADPLL open-loop mode
PLL_FT	11	r / w	512	ADPLL fine-tune value for open-loop mode
PLL_AO	4	r / w	10	ADPLL static coarse tune switch pmos
PLL_SCAP	3	r / w	0	ADPLL static coarse tune switch cap
PLL_SCAP_AUTO_CAL	1	r / w	1	Auto tuning of coarse tune switch cap
PLL_ENFORCE_LOCK	1	r / w	0	Overwrite locked-bit
PLL_PFD_SELECT	1	r / w	0	Enable/disable synchronizer for PFD
PLL_LOCK_WINDOW	1	r / w	1	Lock Detection Window: 0: long; 1: short
PLL_SYNC_BYPASS	1	r / w	0	Bypass lock-in phase synchronization
PLL_FILTER_SHIFT	2	r / w	2	Shift filter coefficients CI and CP left by PLL_FILTER_SHIFT when not locked
PLL_FAST_LOCK	1	r / w	1	Enable internal fast lock-in for fine tune value using binary search
PLL_SAR_LIMIT	3	r / w	0	Limit for binary search fast lock-in
PLL_SET_OP_LOCK	1	r / w	1	Force locked bit to 1 in OP mode
PLL_DISABLE_LOCK	1	r / w	0	Force locked bit to 0
PLL_REF_BYPASS	1	r / w	0	Enable/disable reference clock bypass to ADPLL core clock output
PLL_REF_SEL	1	r / w	0	0: single-ended, 1: LVDS clock
PLL_REF_RTERM	1	r / w	0	Enable / disable reference clock termination
PLL_CONFIG_SEL	1	r / w	1	ADPLL configuration select: 0: use internal values, 1: use configuration from register file
PLL_EN_ADPLL_CTRL	1	r / w	0	Enable / disable ADPLL

2.5.2.3 ADPLL built-in self calibration (BISC)

The ADPLL clock generator supports two integrated calibration schemes for the loop filter proportional gain CP to achieve minimum jitter during operation.

Mode A uses an additional monitor PFD for jitter minimization (not recommended).

Mode B uses the internal PFD signal for jitter minimization (preferred).

Table 2.19: *BISC configuration in the register file*

Register File

Name	Width	Mode	Default	Description
PLL_BISC_MODE	3	r/w	0	xx0: Disable BISC xx1: Enable BISC 00x: Mode A: single shot measurement 01x: Mode A: single shot calibration 10x: Mode B 11x: Mode A: continuous calibration

Table 2.20: *BISC mode A configuration in the register file*

Register File

Name	Width	Mode	Default	Description
PLL_BISC_DLY_PFD_MON_REF	5	r/w	0	
PLL_BISC_DLY_PFD_MON_DIV	5	r/w	2	
PLL_BISC_DLY_DIR	1	r/w	0	
PLL_BISC_PFD_SEL	1	r/w	0	
PLL_BISC_TIMER_MAX	4	r/w	15	
PLL_BISC_CP_MIN	5	r/w	4	
PLL_BISC_CP_MAX	5	r/w	18	
PLL_BISC_CP_START	5	r/w	12	
PLL_BISC_OPT_DET_IND	1	r/w	0	

Table 2.21: *BISC mode B configuration in the register file*

Register File

Name	Width	Mode	Default	Description
PLL_BISC_TIMER_MAX	4	r/w	15	
PLL_BISC_COR_DLY	3	r/w	1	
PLL_BISC_CP_MIN	5	r/w	4	
PLL_BISC_CP_MAX	5	r/w	18	
PLL_BISC_CP_START	5	r/w	12	
PLL_BISC_CAL_SIGN	1	r/w	0	
PLL_BISC_CAL_AUTO	1	r/w	1	

2.5.3 Reset

Table 2.22 shows the interfaces available on the SerDes, which are accessible via the FPGA fabric and can be used to perform various resets within the SerDes architecture.

Table 2.22: *CC_SERDES interfaces for performing various resets*

Name	Width	Direction	Description
PLL_RESET_I	1	Input	Asynchronous ADPLL reset
TX_RESET_I	1	Input	Asynchronous transmitter data path reset
TX_PCS_RESET_I	1	Input	Asynchronous transmitter data path (PCS) reset
TX_PMA_RESET_I	1	Input	Asynchronous transmitter data path (PMA) reset
RX_RESET_I	1	Input	Asynchronous receiver data path reset
RX_PCS_RESET_I	1	Input	Asynchronous receiver data path (PCS) reset
RX_CDR_RESET_I	1	Input	Asynchronous receiver data path (CDR) reset
RX_EQA_RESET_I	1	Input	Asynchronous receiver data path (DFE) reset

The reset signals are asynchronous. During operation, the reset of the ADPLL and the transmitter and receiver data pathes are of importance. For both data paths, the SerDes offers additional interfaces to indicate the current reset status.

For the functioning of the SerDes it is important to know that the resets of the individual components are internally linked. A reset of the transmitter or receiver data path will always trigger a reset of the associated PCS, PMA, CDR or DFE. For the receiver data path this also includes the elastic buffer. A special relationship arises between the state of the ADPLL and the two data paths. The ADPLL keeps the transmitter and receiver data paths in the reset state until a corresponding lock of the ADPLL is present and it is possible to provide the required core clock. A reset of the ADPLL therefore also always results in a reset

Table 2.23: *Interface for the loopback status in the register file*

Name	Width	Direction	Description
TX_RESET_DONE_O	1	Output	Reset status of the transmitter data path. The output changes from 0 to 1 when the reset is completed. The signal remains active 1 until a new reset is triggered.
RX_RESET_DONE_O	1	Output	Reset status of the receiver data path. The output changes from 0 to 1 when the reset is completed. The signal remains active 1 until a new reset is triggered.

of both data paths. This results for the FPGA reset process and the associated configuration of the SerDes automatic chaining of the resets. This starts with the activation of the ADPLL via the register field PLL_EN_ADPLL_CTRL (address 0x 50 [0]) and is completed with the edge changes of the status signals TX_RESET_DONE_O and RX_RESET_DONE_O. It should also be noted that the receiver data path reaches its reset state with a time delay after the transmitter data path.

2.5.4 Loopback

Loopback testing allows for self-diagnosis and verification of the SerDes functionality without external dependencies. It validates the integrity of data transmission and reception within the internal circuits.

Near-end loopback refers to a loopback where the transmitted signal is fed back into the receiver at the same device, on the same side of the communication link (i.e., on the sending side). In an FPGA context, this would involve taking the output from a SerDes and routing it back into the input of the same SerDes (or another receiver block within the same FPGA). This allows to test the transmitter and receiver functionality without needing a physical connection to another device.

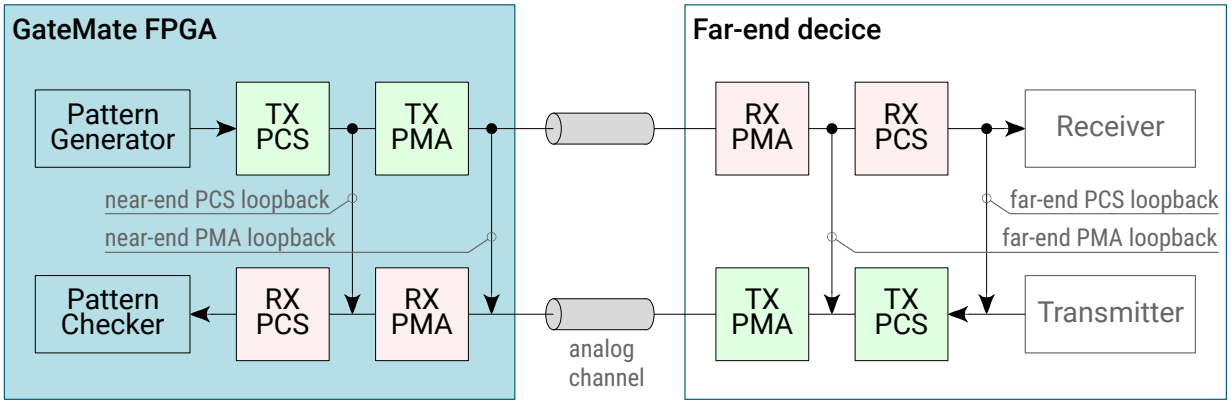


Figure 2.32: *Loopback terminology*

Table 2.24: *Interface for the activation of the loopback mode*

Name	Width	Direction	Description
LOOPBACK_I	3	Input	Loopback mode selection 000: Normal operation 001: Near-end PCS loopback 010: Near-end PMA loopback 011: Reserved 100: Far-end PMA loopback 101: Reserved 110: Far-end PCS loopback 111: Reserved

Table 2.25: *Loopback configuration in the register file*

Register File

Name	Width	Mode	Default	Description
TX_LOOPBACK_OVR	1	r/w	0	Transmitter loopback overwrite when SerDes is in testmode.
TX_PMA_LOOPBACK	2	r/w	0	Force transmitter near-end PMA loopback when TX_LOOPBACK_OVR is active. If overwrite is not active, set transmitter PMA loopback mode. TX_PMA_LOOPBACK[1] must be 1'b0. Set to 2'b01 for near-end PMA loopback.
TX_PCS_LOOPBACK	1	r/w	0	Force transmitter near-end PCS loopback when TX_LOOPBACK_OVR is active.
RX_LOOPBACK_OVR	1	r/w	0	Receiver loopback overwrite when SerDes is in testmode.
RX_PMA_LOOPBACK	1	r/w	0	Force receiver far-end PMA loopback when RX_LOOPBACK_OVR is active.
RX_PCS_LOOPBACK	1	r/w	0	Force receiver far-end PCS loopback when RX_LOOPBACK_OVR is active.

Far-end loopback, on the other hand, involves sending data from the transmitter at one end (e.g., one FPGA or device) across the physical link to the receiver at the far end (e.g., another FPGA or device) and then looping the signal back to the original transmitter. In the context of an FPGA, the far-end loopback requires communication over a physical link (e.g., over a high-speed serial connection) to the other side, where the signal is received and then fed back into the transmitter of the far-end device.

- **Near-end PCS loopback:**
  - Redirects the transmitted data back to the receiver within the PCS before the PMA to verify the integrity of the PCS layer and the subsequent stages of transmission.
- **Near-end PMA loopback:**
  - Redirects the transmitted data back to the receiver within the PMA layer to validate the integrity of the PMA layer and the physical transmission characteristics.
- **Far-end PMA loopback:**
  - Redirects the transmitted data back to the receiver after traversing the entire communication channel and reaching the end-stage PMA to validate end-to-end physical transmission characteristics, including external components and PMA behavior.
- **Far-End PCS Loopback:**
  - Redirects the transmitted data back to the receiver after traversing the entire communication channel to validate end-to-end data integrity, including transmission through external components.

To activate a loopback, the corresponding coding can be applied to the input `LOOPBACK_I`. See Table 2.24 and Figure 2.29 for reference.

2.5.5 Transceiver Interface

This section describes the SerDes transmitter interface and its features.

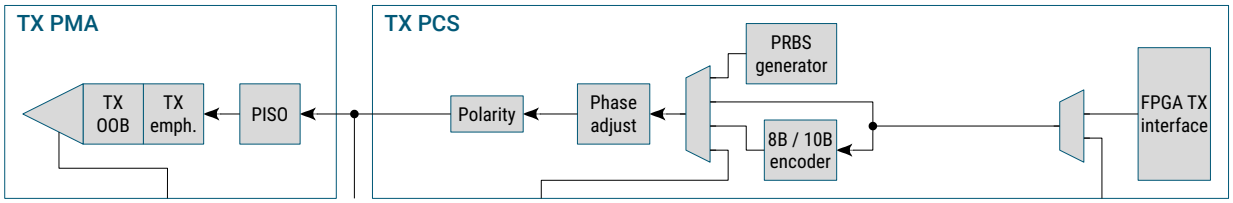


Figure 2.33: *SerDes transmitter path overview*

2.5.5.1 Transmitter Data Path

The transmitter interface is the direct connection to the FPGA fabric. The width of the data path can be configured using the `TX_DATAPATH_SEL` (regfile `0x40[4:3]`) parameter. Port widths can be 16 / 20, 32 / 40 or 64 / 80 bits. With no 8B / 10B encoder enabled, the 64-bit wide input `TX_DATA_I` is extended with the 8-bit wide inputs `TX_CHAR_DISPMODE_I` and `TX_CHAR_DISPVAL_I` each. The clock rate is determined by the transmitter line rate, the data path width and the 8B / 10B encoder. 8B / 10B encoding is described in more detail in Section 2.5.5.2.

Table 2.26 shows the related transmitter interface ports.

Table 2.26: *CC_SERDES transmitter data path related ports*

Name	Width	Direction	Description
`TX_CLK_I`	1	Input	Transmitter data path clock, may come from the PLL's `CLK_CORE_O` output.
`TX_RESET_I`	1	Input	Asynchronous transmitter data path reset.
`TX_PCS_RESET_I`	1	Input	Asynchronous transmitter PCS reset.
`TX_RESET_DONE_O`	1	Output	Transmitter reset state indicator.
`TX_DATA_I`	64	Input	Transmitter input data bus.
`TX_CHAR_DISPMODE_I`	8	Input	Data bus extension if 8B / 10B encoder is not used. Refer to Figure 2.34 for detailled information.
`TX_CHAR_DISPVAL_I`	8	Input	Data bus extension if 8B / 10B encoder is not used. Refer to Figure 2.34 for detailled information.

With no 8B / 10B encoder enabled, the transmitter data path ordering is as illustrated in Figure 2.34.

Table 2.27: *Transmitter configuration in the register file*

Register File

Name	Width	Mode	Default	Description
TX_DATAPATH_SEL	2	r / w	3	Transmitter datapath width selection: 00: 16 bit / 20 bit 01: 32 bit / 40 bit 1X: 64 bit / 80 bit
TX_DATA_OVR	1	r / w	0	Transmitter data overwrite enable when SerDes is in testmode.
TX_DATA_CNT	3	r / w	0	Transmitter data overwrite pointer; indicates next 16 bit word to read or write during access to TX_DATA.
TX_DATA_VALID	1	r / w	0	Transmitter data valid indicator when TX_DATA_OVR is active.
TX_DATA	16	r / w	0	Transmitter data value to set with current TX_DATA_CNT; auto-increments TX_DATA_CNT by read or write access; wrap-around occurs automatically depending on TX_DATAPATH_SEL selection.
TX_PCS_RESET_OVR	1	r / w	0	Transmitter PCS reset overwrite when SerDes is in testmode.
TX_PCS_RESET	1	r / w	0	Transmitter PCS reset value when TX_PCS_RESET_OVR is active.
TX_PCS_RESET_TIME	5	r / w	3	Number of CLK_CORE_TX_I clock cycles for that internal PCS reset should be active.
TX_RESET_OVR	1	r / w	0	Transmitter reset overwrite when SerDes is in testmode.
TX_RESET	1	r / w	0	Transmitter reset value when TX_RESET_OVR is active.

Table 2.28: *Transmitter interface status in the register file*

Register File

Name	Width	Mode	Default	Description
TX_DATAPATH_SEL	2	R/W	3	Transmitter datapath width selection: 00: 16 bit / 20 bit 01: 32 bit / 40 bit 1X: 64 bit / 80 bit
TX_RESET_DONE	1	r	0	Transmitter reset status.

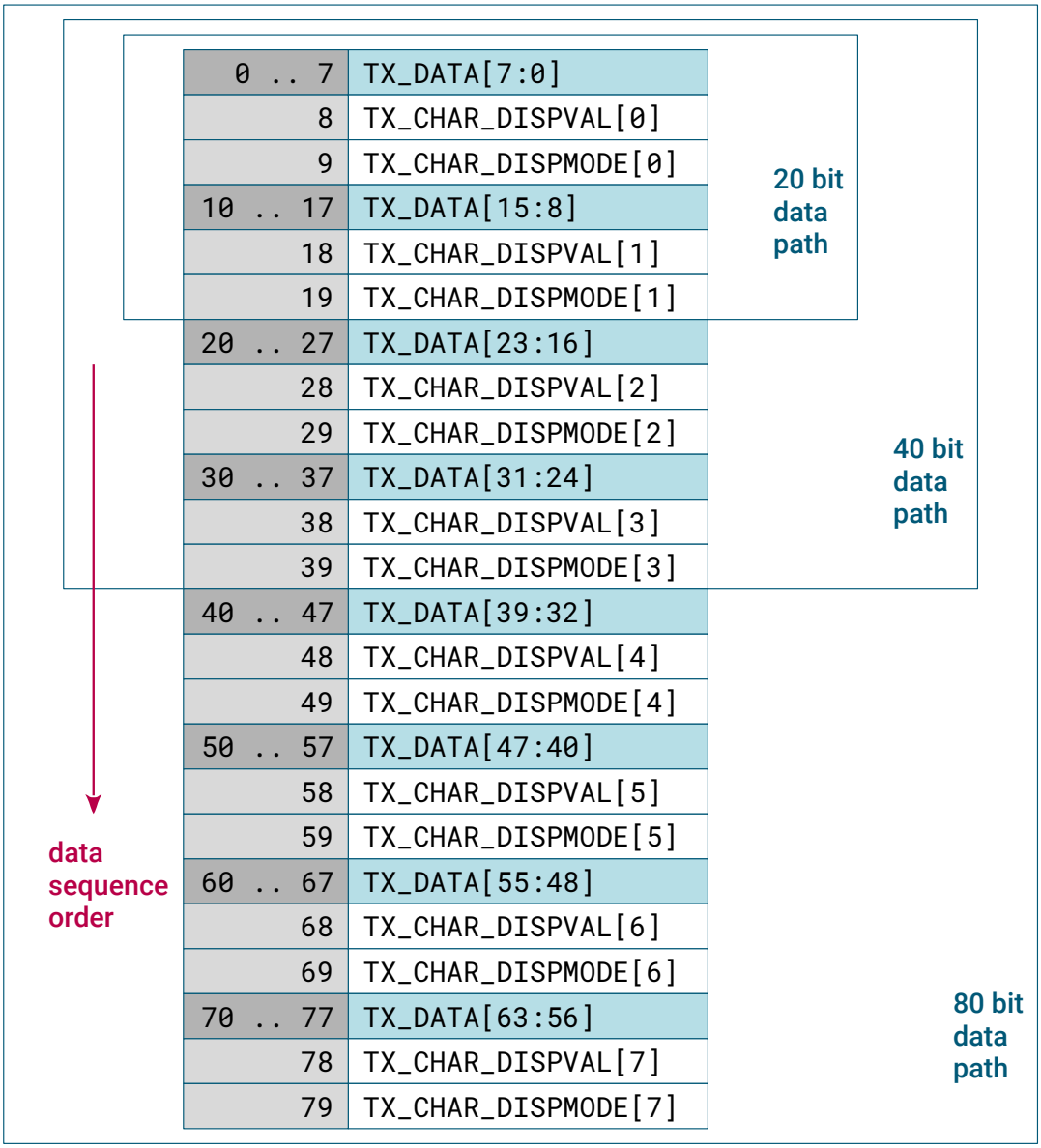


Figure 2.34: *Transmitter data ordering with no 8B/10B encoding (20-, 40- or 80-Bit datapath)*

2.5.5.2 Transmitter 8B /10B Encoder

The SerDes has a build-in 8B / 10B encoder. It is a widely-used technique in high-speed data transmission to ensure reliable and efficient communication. It converts 8-bit data into a 10-bit code by adding extra bits for error detection and maintaining a balance between 0s and 1s to facilitate clock recovery. This encoding scheme helps in reducing DC component, ensuring signal integrity, and enabling easy synchronization between transmitter and receiver in high-speed communication systems.

Table 2.29: *CC_SERDES interface for the 8B / 10B encoder*

Name	Width	Direction	Description
TX_8B10B_BYPASS_I	8	Input	Each high bit bypasses the corresponding TX_DATA_I byte. Refer to Table 2.31 for detailled information.
TX_8B10B_EN_I	1	Input	Enable transmitter 8B / 10B encoder.
TX_CHAR_DISPMODE_I	8	Input	Overrides running disparity according to value set in TX_CHAR_DISPVAL_I. Used as data bus extension if 8B / 10B encoder is not used. Refer to Table 2.30 for detailled information.
TX_CHAR_DISPVAL_I	8	Input	Provides running disparity control. Used as data bus extension if 8B / 10B encoderis not used. Refer to Table 2.30 for detailled information.
TX_CHAR_IS_K_I	8	Input	If high corresponding TX_DATA_I byte is interpreted as a valid K character. Refer to Table 2.31 for detailed information.

Table 2.30: *Disparity behavior*

DISPMODE	DISPVAL	Disparity
0	0	Use build-in 8B / 10B disparity calculation
0	1	Invert disparity
1	0	Force negative disparity
1	1	Force positive disparity

Enabling the 8B / 10B encoder increases latency through the transmitter path. However, it is possible to disable and bypass the 8B / 10B encoder to minimize latency, if it is not needed.

2.5.5.3 Transmitter FIFO

The SerDes Trensceiver has a build-in phase-adjust FIFO or transmit buffer to resolve phase differences between the transmitter PCS and PMA domains.

Table 2.31: *Corresponding data byte on TX_DATA for TX_8B10B_BYPASS or TX_CHAR_IS_K*

TX_8B10B_BYPASS_I bit	TX_CHAR_IS_K_I bit	TX_DATA byte
TX_8B10B_BYPASS_I[0]	TX_CHAR_IS_K_I[0]	TX_DATA[7:0]
TX_8B10B_BYPASS_I[1]	TX_CHAR_IS_K_I[1]	TX_DATA[15:8]
TX_8B10B_BYPASS_I[2]	TX_CHAR_IS_K_I[2]	TX_DATA[23:16]
TX_8B10B_BYPASS_I[3]	TX_CHAR_IS_K_I[3]	TX_DATA[31:24]
TX_8B10B_BYPASS_I[4]	TX_CHAR_IS_K_I[4]	TX_DATA[39:32]
TX_8B10B_BYPASS_I[5]	TX_CHAR_IS_K_I[5]	TX_DATA[47:40]
TX_8B10B_BYPASS_I[6]	TX_CHAR_IS_K_I[6]	TX_DATA[55:48]
TX_8B10B_BYPASS_I[7]	TX_CHAR_IS_K_I[7]	TX_DATA[64:56]

Table 2.32: *CC_SERDES interface for the transmit buffer*

Name	Width	Direction	Description
TX_BUF_ERR_O	1	Output	High in case of a buffer over- or underflow. Is cleared with TX_RESET_I and RESET_CORE_TX_N_I.

Table 2.33: *Transmit buffer status in the register file*

Register File

Name	Width	Mode	Default	Description
TX_BUF_ERR	1	r	0	High in case of a buffer over- or underflow. Is cleared with TX_RESET_I and RESET_CORE_TX_N_I.

2.5.5.4 Transmitter Pattern Generator

The SerDes incorporates a build-in pattern generator block designed to generate various industry-standard pseudo-random bit stream (PRBS), 2-UI or 20 / 40 / 80 UI square wave test patterns. These sequences or patterns serve as valuable tools for assessing the signal integrity of high-speed connections. While they exhibit random-like behavior, these sequences possess distinct characteristics crucial for evaluating link quality and performance.

Table 2.34: *CC_SERDES interface for the pattern generator*

Name	Width	Direction	Description
TX_PRBS_SEL_I	3	Input	Transmitter PRBS mode selection 000 Bypass PRBS generator 001 PRBS-7 010 PRBS-15 011 Reserved 100 Reserved 101 Reserved 110 2 UI square wave (alternating 0s / 1s) 111 20 / 40 / 80 UI square wave
TX_PRBS_FORCE_ERR_I	1	Input	Injects errors in the transmitted PRBS sequence if driven high

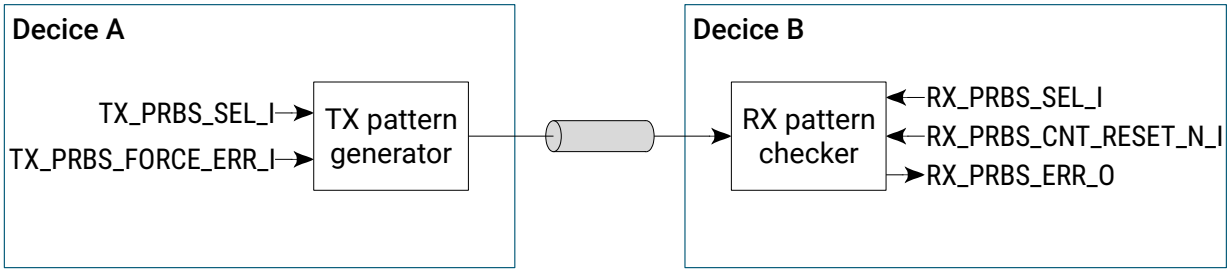


Figure 2.35: *SerDes link quality test*

In combination as shown in Figure 2.35, transmitter pattern generator and receiver pattern checker can be used to provide information on link quality or jitter tolerance. For further information on the receiver pattern checker see Section 2.5.6.6.

2.5.5.5 Transmitter Polarity Control

The transmitter interface allows inversion of the outgoing differential data to match the PCB signals. Driving TX_POLARITY_I high enables inversion of the polarity of outgoing data.

Table 2.35: *CC_SERDES transmitter polarity control*

Name	Width	Direction	Description
TX_POLARITY_I	1	Input	Transmitter polarity inversion control 0: Normal operation 1: Invert polarity of outgoing data stream

2.5.5.6 Transmitter Configurable Driver

The electrical line driver utilizes a switched current mirror architecture, with multiple scaling factors that collectively determine the final differential output swing and common-mode voltage. The driver consists of 125 selectable branches, each of which can be individually activated to contribute to the output signal. Non-activated branches remain inactive and do not influence the output swing.

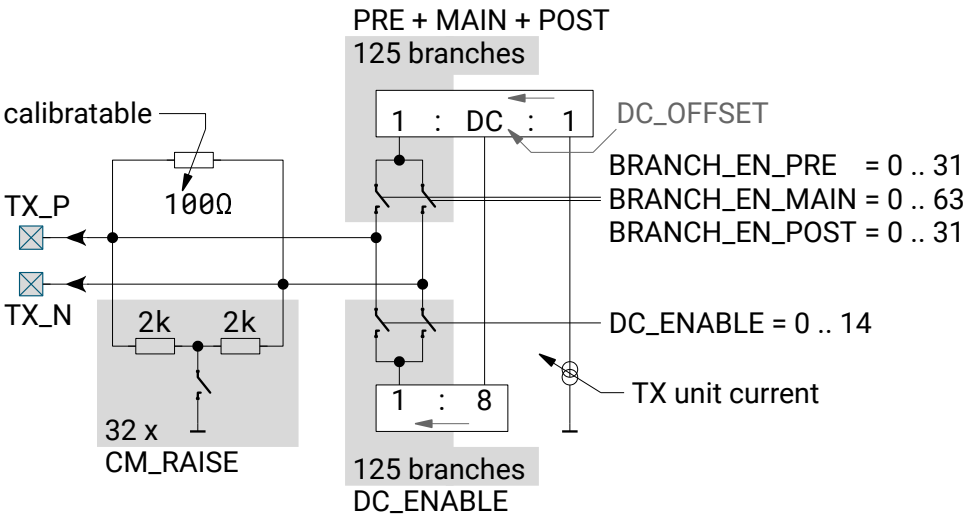


Figure 2.36: *Simplified SerDes transmitter driver*

These branches are organized into three groups to facilitate feed-forward equalization (FFE), commonly referred to as de-emphasis and pre-emphasis, on the transmitter side. The first group contains 63 branches, which can either contribute to the main cursor or remain off. The second group includes 31 branches, which can contribute to either the main or pre-cursors, or remain off. The third group also consists of 31 branches, capable of contributing to either the main or post-cursor, or remaining off.

The differential output swing is developed by the drive current flowing through the effective resistance, which is determined by the parallel combination of the transmitter termination and line impedance—typically $50\,\Omega$ differentially. By adjusting the number of active branches and their assignment to pre-, main-, or post-cursors, de-emphasis and pre-emphasis can be effectively realized.

Table 2.36: *DFE buffer configuration in the register file*

Register File

Name	Width	Mode	Default	Description
TX_SEL_PRE	5	r/w	0	Number of pre-cursor branches, must not exceed TX_BRANCH_EN_PRE.
TX_SEL_POST	5	r/w	0	Number of post-cursor branches, must not exceed TX_BRANCH_EN_POST.
TX_AMP	5	r/w	15	Tranmit unit current.
TX_BRANCH_EN_PRE	5	r/w	0	Number of active pre/main branches.
TX_BRANCH_EN_MAIN	6	r/w	0x3F	Number of active main branches.
TX_BRANCH_EN_POST	5	r/w	0	Number of active post/main branches.
TX_TAIL_CASCODE	3	r/w	4	Control voltage for transmitter unit current. **Keep default.**
TX_DC_ENABLE	7	r/w	63	Number of active branches for common mode adjustment.
TX_DC_OFFSET	5	r/w	0	Additional scaling for common mode adjustment. **Set to 8.**
TX_CM_RAISE	5	r/w	0	Adjustment for common mode voltage.
TX_CM_THRESHOLD_0	5	r/w	14	Lower setpoint for common mode regulation.
TX_CM_THRESHOLD_1	5	r/w	16	Setpoint for common mode regulation.

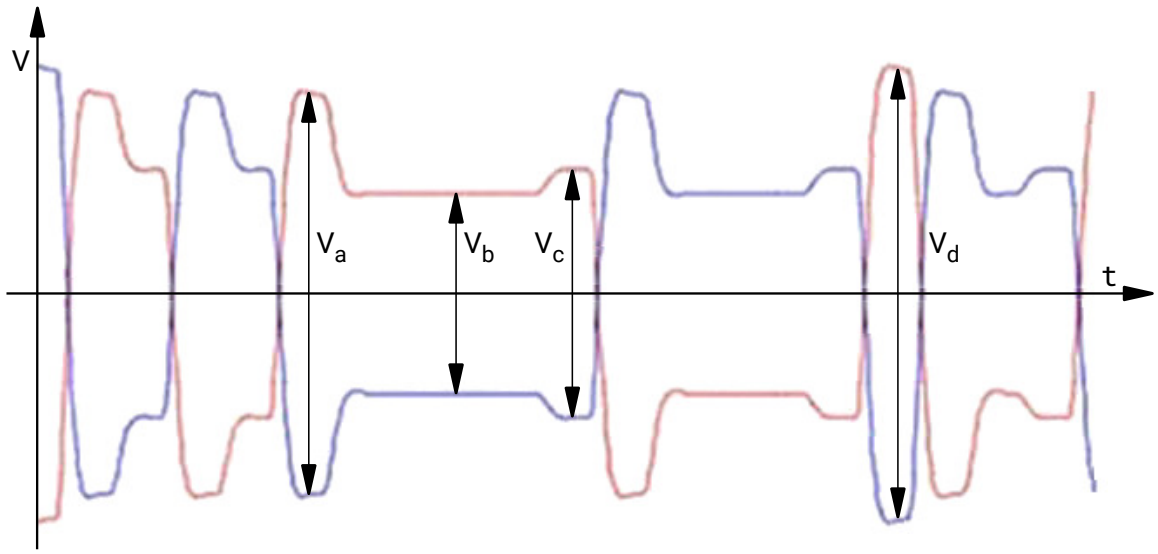


Figure 2.37: *Simplified transmitter feed-forward equalization (FFE) scheme for signal (V_a), de-emphasized (V_b), pre-emphasized (V_c) and boost (V_d) voltages*

The following formulae summarize the configuration options for branch assignment:

$$I_{TX} = (\texttt{TX\_TAIL\_CASCODE} + 10) \cdot (\texttt{TX\_AMP} + 1) \cdot 9.375\,\mu A \tag{2.6}$$

$$N_{BRA} = \texttt{TX\_BRANCH\_EN\_PRE} + \texttt{TX\_BRANCH\_EN\_MAIN} + \texttt{TX\_BRANCH\_EN\_POST} \tag{2.7}$$

$$\begin{aligned} N_{PRE\_CUR} &= \texttt{TX\_SEL\_PRE} \\ N_{MAIN\_CUR} &= N_{BRA} - N_{PRE\_CUR} - N_{POST\_CUR} \\ N_{POST\_CUR} &= \texttt{TX\_SEL\_POST} \end{aligned} \tag{2.8}$$

$$\begin{aligned} V_a &= (N_{MAIN\_CUR} - N_{PRE\_CUR} + N_{POST\_CUR}) \cdot I_{TX} \cdot 50\,\Omega \\ V_b &= (N_{MAIN\_CUR} - N_{PRE\_CUR} - N_{POST\_CUR}) \cdot I_{TX} \cdot 50\,\Omega \\ V_c &= (N_{MAIN\_CUR} + N_{PRE\_CUR} - N_{POST\_CUR}) \cdot I_{TX} \cdot 50\,\Omega \\ V_d &= (N_{MAIN\_CUR} + N_{PRE\_CUR} + N_{POST\_CUR}) \cdot I_{TX} \cdot 50\,\Omega = N_{BRA} \cdot 50\,\Omega \end{aligned} \tag{2.9}$$

Common mode voltage is adjusted as a combination of TX_CM_RAISE adjusting the common-mode contribution of the termination and TX_DC_OFFSET to shift the common mode further. A digital regulation loop is implemented which when enables controls common mode pull-down branch number other controlled through TX_DC_OFFSET. The target for the regulation loop is set by TX_CM_THRESHOLD_0 and TX_CM_THRESHOLD_1, which define a window of valid voltages for the regulation:

$$V_{DD} \cdot \frac{14 + \texttt{TX\_CM\_THRESHOLD\_0}}{60} \leqslant V_{CM} \leqslant V_{DD} \cdot \frac{14 + \texttt{TX\_CM\_THRESHOLD\_1}}{60} \tag{2.10}$$

The digital common mode regulator is a PI regulator, whose filter coefficients can be adjusted. Common mode voltage can be measured with two simple SAR ADCs, when digital common mode regulation is disabled by setting

$$\texttt{TX\_CM\_THRESHOLD\_0} = \texttt{TX\_CM\_THRESHOLD\_1} \quad .$$

The results correspond to the following formula:

$$V_{CM} = V_{DD} \cdot \frac{14 + \texttt{TX\_CM\_THRESHOLD\_0}}{60} \tag{2.11}$$

Transmitter Electrical Idle

All settings exist in a duplicate version for electrical idle mode. Reduced number of branches allows for power reduction in electrical idle mode. Electrical idle settings are in effect in electrical idle mode. Names of config fields and registers are as those for normal operation with _EI appended.

Transmitter Receiver Detection

Far end receiver detection can be accomplished by assigning a different common mode level to the driver and observing the actual current mode voltage on the pair of lines. A third set of all configuration settings is in effect for the duration of the modified common mode output.

2.5.6 Receiver Interface

This section describes the SerDes receiver interface and its features.

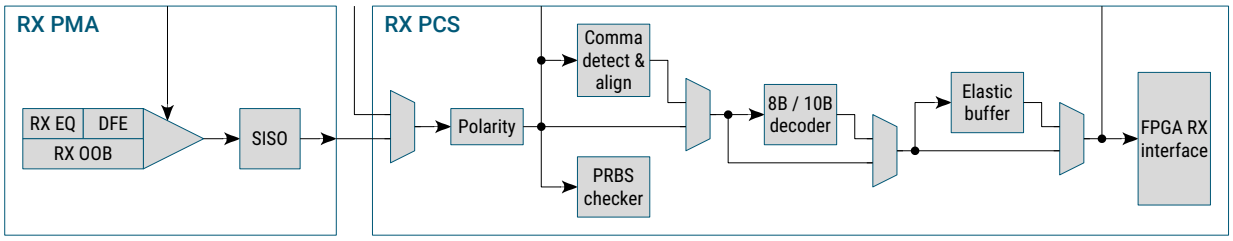


Figure 2.38: *SerDes receiver path overview*

2.5.6.1 Receiver Data Path

The receiver interface is the direct connection to the FPGA fabric. The width of the data path can be configured using the RX_DATAPATH_SEL parameter. Port widths can be 16 / 20, 32 / 40 or 64 / 80 bits. With no 8B / 10B decoder enabled, the 64-bit wide input RX_DATA_O port is extended with the 8-bit wide outputs RX_CHAR_IS_K_O and RX_DISP_ERR_O. The data path clock PLL_CLK_O frequency is determined by the line rate, the data path width and the 8B / 10B decoder. 8B / 10B decoding is described in more detail in Section 2.5.6.7.

Table 2.37 shows the related receiver interface ports.

Table 2.37: *CC_SERDES receiver data path related ports*

Name	Width	Direction	Description
RX_CLK_I	1	Input	Receiver data path clock, may come from the PLL's PLL_CLK_O or RX_CLK_O outputs.
RX_CLK_O	1	Output	Receiver recovered clock.
RX_RESET_I	1	Input	Asynchronous receiver data path reset.
RX_PCS_RESET_I	1	Input	Asynchronous receiver PCS reset.
RX_RESET_DONE_O	1	Output	Receiver reset state indicator.
RX_DATA_O	64	Output	Receiver output data bus.
RX_CHAR_IS_K_O	8	Output	Data bus extension if 8B / 10B encoder is not used. Refer to Figure 2.39 for detailled information.
RX_DISP_ERR_O	8	Input	Data bus extension if 8B / 10B encoderis not used. Refer to Figure 2.39 for detailled information.

With no 8B / 10B decoder enabled, the receiver data path ordering is as illustrated in Figure 2.39.

Table 2.38: *Receiver configuration in the register file*

Register File

Name	Width	Mode	Default	Description
RX_DATAPATH_SEL	2	r/w	3	Receiver datapath width selection: 00: 16 bit / 20 bit 01: 32 bit / 40 bit 1x: 64 bit / 80 bit
RX_PCS_RESET_OVR	1	r/w	0	Receiver PCS reset overwrite when SerDes is in testmode
RX_PCS_RESET	1	r/w	0	Receiver PCS reset value when RX_PCS_RESET_OVR is active
RX_PCS_RESET_TIME	5	r/w	3	Number of RX_CLK_I clock cycles prescaled with RX_RESET_TIMER_PRESC for internal PCS reset should be active
RX_RESET_OVR	1	r/w	0	Receiver reset overwrite when SerDes is in testmode
RX_RESET	1	r/w	0	Receiver reset value when RX_RESET_OVR is active
RX_RESET_DONE_GATE	1	r/w	0	Prevent RX_RESET_DONE_O to indicate successful reset cycle completion. Only used for debugging purposes

Table 2.39: *Receiver interface status in the register file*

Register File

Name	Width	Mode	Default	Description
RX_RESET_DONE	1	r	0	Receiver reset status.

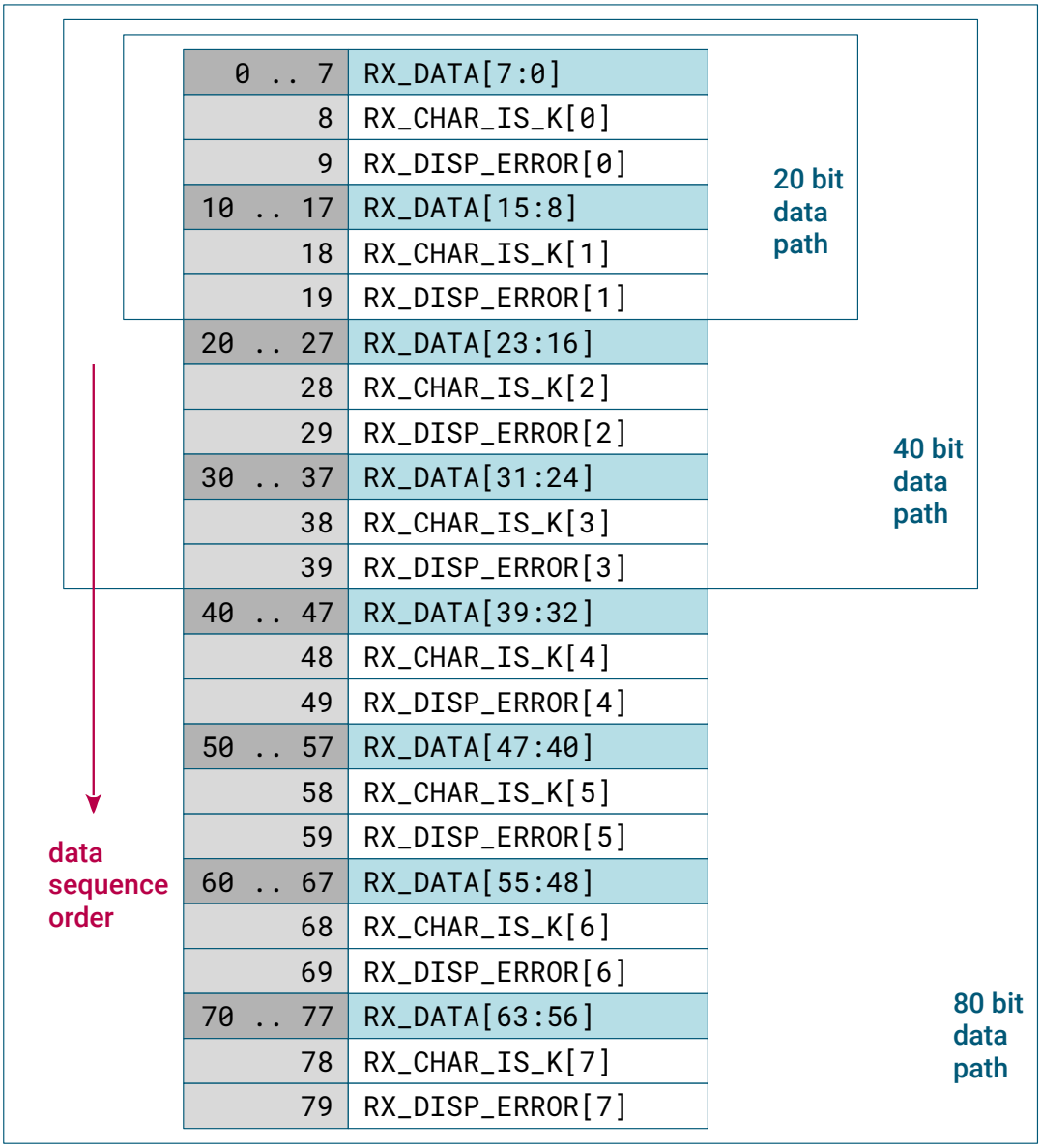


Figure 2.39: *Receiver data ordering with no 8B / 10B encoding (20-, 40- or 80-Bit datapath)*

2.5.6.2 Receiver Analog Front End

The receiver analog front end is a high-speed current-mode differential input buffer. It features configurable receiver termination voltages, calibratable termination resistors and an electrical idle detector.

Table 2.40: *PMA function configuration in the register file*

Register File

Name	Width	Mode	Default	Description
RX_RESET_TIMER_PRESC	5	r/w	0	Receiver reset timer prescaler.
RX_PMA_RESET_OVR	1	r/w	0	Receiver PMA reset overwrite when SerDes in testmode.
RX_PMA_RESET	1	r/w	0	Receiver PMA reset value when RX_PMA_RESET_OVR active.
RX_PMA_RESET_TIME	5	r/w	3	Number of clk_core_rx_i cycles prescaled with RX_RESET_TIMER_PRESC for active PMA reset.
RX_POWER_DOWN_OVR	1	r/w	0	Receiver power down overwrite when SerDes in testmode.
RX_POWER_DOWN_N	1	r/w	0	Receiver power down value when RX_POWER_DOWN_OVR is active.

2.5.6.3 Receiver Equalizer

The SerDes has a build-in 3-tap decision feedback equalizer (DFE). The primary purpose of a DFE is to mitigate the effects of intersymbol interference (ISI) in a communication channel. ISI occurs when neighboring symbols in a digital communication signal start to overlap due to various factors such as signal distortion, noise, or bandwidth limitations. This overlapping can make it challenging to accurately detect the transmitted symbols.

The DFE works by using feedback from previously detected symbols to adjust and equalize the incoming signal, aiming to reconstruct the original transmitted data. The “3-tap” in a 3-tap DFE refers to the number of adjustable delay elements (or taps) used in the equalization process.

Each tap in the DFE represents a coefficient that adjusts the incoming signal based on the detected symbols. By considering multiple taps, the DFE can better compensate for the distortion and effectively “look ahead” and “look back” at adjacent symbols to improve signal quality.

Table 2.41: *PMA buffer configuration in the register file*

Register File

Name	Width	Mode	Default	Description
RX_AFE_PEAK	5	r/w	16	Receiver set peaking in AFE (0 = max, 31 = min).
RX_AFE_GAIN	4	r/w	8	Receiver set gain in AFE (0 = min, 15 = max).
RX_AFE_VCMSEL	3	r/w	4	Receiver set common mode level of internal receive AFE receiver chain.
RX_CALIB_EN	1	r/w	0	Receiver termination resistor calibration start enable.
RX_CALIB_OVR	1	r/w	0	Receiver termination resistor calibration overwrite.
RX_CALIB_VAL	4	r/w	0	Receiver termination resistor calibration value for overwrite.
RX_RTERM_VCMSEL	3	r/w	4	Receiver set common mode level; is applied through RTERM to the line. Value is $(18..25)/29*\text{VDDIO} = (0.62..0.86)*\text{VDDIO}$. In AC mode this ist the input common mode of the receive amplifier. In case of a small input signal higher CM is suggested. In DC mode the common mode is defined by the transmitter.
RX_RTERM_PD	1	r/w	0	
RX_CALIB_DONE	1	r	1	Receiver termination resistor calibration done indicator.
RX_CALIB_CAL	4	r	0	Receiver termination resistor calibration result.

Table 2.42: *Electrical idle (EI) configuration in the register file*

Register File

Name	Width	Mode	Default	Description
RX_EI_BIAS	4	r/w	4	Receiver EI bias value[4:1].
RX_EI_BW_SEL	4	r/w	4	Receiver EI bandwidth selection.
RX_EN_EI_DETECTOR_OVR	1	r/w	0	Receiver electrical idle detector enable overwrite when SerDes in testmode.
RX_EN_EI_DETECTOR	1	r/w	0	Receiver electrical idle detector enable value if RX_EN_EI_DETECTOR_OVR active.

Table 2.43: *DFE function configuration in the register file*

Register File

Name	Width	Mode	Default	Description
RX_EQA_RESET_OVR	1	r/w	0	Receiver equalizer (DFE) reset overwrite when SerDes is in testmode.
RX_EQA_RESET	1	r/w	0	Receiver equalizer reset value when RX_EQA_RESET_OVR is active.
RX_EQA_RESET_TIME	5	r/w	3	Number of CLK_CORE_RX_I clock prescaled with RX_RESET_TIMER_PRESC cycles for that internal equalizer reset should be active
RX_EQA_CKP_LF	8	r/w	0x A3	Proportional coefficient of control loop for low frequency amplitude detection
RX_EQA_CKP_HF	8	r/w	0x A3	Proportional coefficient of control loop for high frequency amplitude detection
RX_EQA_CKP_OFFSET	8	r/w	0x 01	Proportional coefficient of control loop for offset detection
RX_EN_EQA	1	r/w	0	Enable DFE adaption
RX_EQA_LOCK_CFG	1	r/w	0	Lock configuration of DFE adaption
RX_TH_MON1	5	r/w	8	Set threshold of monitor 1
RX_TH_MON2	5	r/w	8	Set threshold of monitor 2
RX_TAPW	5	r/w	8	Set DFE tap weight
RX_AFE_OFFSET	5	r/w	8	Set AFE offset
RX_EN_EQA_VALUE	4	r/w	0	Enable overriding of Bit 0: tap monitor 1 Bit 1: tap monitor 2 Bit 2: tap weight Bit 3: AFE offset (0 = disable, 1 = enable)
RX_EQA_CONFIG	16	r/w	0x 01 C0	Configuration of DFE adaption
RX_MON_PH_OFFSET	6	r/w	0	Define phase offset of monitor 2

Table 2.44: *DFE status in the register file*

Register File

Name	Width	Mode	Default	Description
RX_EQA_LOCKED	1	r	0	DFE lock status
RX_EQA_TAPW	1	r	0	DFE tap weight
RX_TH_MON	1	r	0	Threshold applied to monitor 2
RX_OFFSET	1	r	0	Offset value for AFE

Receiver Margin Analysis

Receiver margin analysis is essential for assessing the reliability of high-speed serial data interfaces in FPGAs. As data rates increase, it's crucial to ensure the receiver can handle signal variations such as jitter and noise. This analysis helps identify how well the receiver performs under different conditions, ensuring robust data integrity.

Table 2.45: *Eye measurement configuration in the register file*

Register File

Name	Width	Mode	Default	Description
RX_EYE_MEAS_EN	1	r/w	0	Start eye measurement
RX_EYE_MEAS_CFG	15	r/w	0	Eye measurement configuration

Eye measurement is a key tool in this process, providing a visual representation of signal quality through an eye diagram. This diagram reveals critical parameters like eye height and width, helping engineers evaluate and optimize receiver performance.

Receiver Clock Data Recovery (CDR)

The SerDes incorporates a Clock and Data Recovery (CDR) feature for synchronizing incoming data streams with the receiver's internal clock.

2.5.6.4 Receiver Polarity Control

The receiver interface allows inversion of the incoming differential data to match the PCB signals. Driving RX_POLARITY_I high enables inversion of the polarity of incoming data.

Table 2.46: *Eye measurement status in the register file*

Register File

Name	Width	Mode	Default	Description
RX_EYE_MEAS_CORRECT_11S	16	r	0	Number of correct detected 0b X11 sequences
RX_EYE_MEAS_WRONG_11S	16	r	0	Number of wrong detected 0b X11 sequences
RX_EYE_MEAS_CORRECT_00S	16	r	0	Number of correct detected 0b X00 sequences
RX_EYE_MEAS_WRONG_00S	16	r	0	Number of wrong detected 0b X00 sequences
RX_EYE_MEAS_CORRECT_001S	16	r	0	Number of correct detected 0b 001 sequences
RX_EYE_MEAS_WRONG_001S	16	r	0	Number of wrong detected 0b 001 sequences
RX_EYE_MEAS_CORRECT_110S	16	r	0	Number of correct detected 0b 110 sequences
RX_EYE_MEAS_WRONG_110S	16	r	0	Number of wrong detected 0b 110 sequences

Table 2.47: *CDR configuration in the register file*

Register File

Name	Width	Mode	Default	Description
RX_CDR_RESET_OVR	1	r/w	0	Receiver CDR reset overwrite when SerDes is in testmode
RX_CDR_RESET	1	r/w	0	Receiver CDR reset value when RX_EQA_RESET_OVR is active
RX_CDR_RESET_TIME	5	r/w	3	Number of CLK_CORE_RX_I clock prescaled with RX_RESET_TIMER_PRESC cycles for that internal CDR reset should be active
RX_WAIT_CDR_LOCK	1	r/w	1	If active, then reset sequence is stopped, until CDR lock is achieved
RX_CDR_CKP	8	r/w	0x F8	Proportional coefficient of control loop
RX_CDR_CKI	8	r/w	0	Integral coefficient of control loop
RX_CDR_TRANS_TH	9	r/w	128	Min. no. of PFD transitions
RX_CDR_LOCK_CFG	6	r/w	0x 0B	CDR lock configuration
RX_CDR_FREQ_ACC	15	r/w	0	Frequency accu value to be set
RX_CDR_PHASE_ACC	16	r/w	0	Phase accu value to be set
RX_CDR_SET_ACC_CONFIG	2	r/w	0	Enable overriding frequency (bit 1) and phase (bit 0) values
RX_CDR_FORCE_LOCK	1	r/w	0	Force CDR lock
RX_CDR_LOCKED	1	r	0	Read lock status signal
RX_CDR_FREQ_ACC_VAL	16	r	0	Read freq accu value
RX_CDR_PHASE_ACC_VAL	16	r	0	Read phase accu value

Table 2.48: *CC_SERDES receiver polarity control*

Name	Width	Direction	Description
RX_POLARITY_I	1	Input	Receiver polarity inversion control 0: Normal operation 1: Invert polarity of incoming data stream

2.5.6.5 Receiver Byte and Word Alignment

Receiver byte and word alignment ensures accurate data interpretation by aligning serialized data to the correct byte and word boundaries. In high-speed serial interfaces, data is transmitted as a continuous stream of bits, which must be properly aligned for accurate parallel processing.

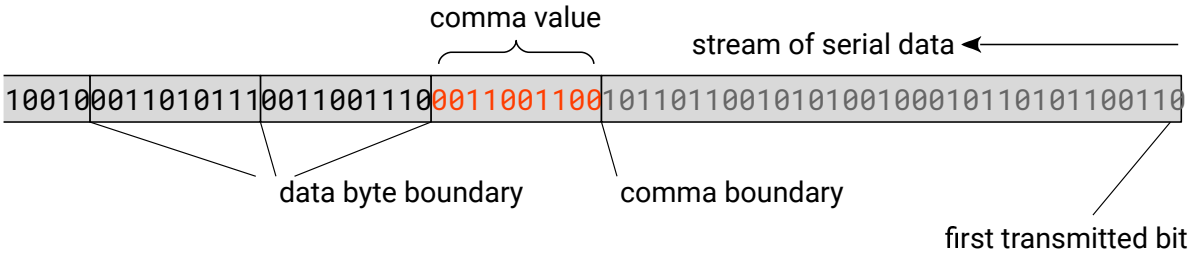


Figure 2.40: *Aligning to a 10-bit comma*

Table 2.49: *CC_SERDES interface for the byte and word alignment*

Name	Width	Direction	Description
RX_BYTE_IS_ALIGNED_O	1	Output	High if comma detection has aligned incoming commas to the byte boundaries successfully.
RX_BYTE_REALIGN_O	1	Output	High if alignment circuit has detected an unaligned comma and realigned it to the new boundary. If signal turns from high to low the circuit has aligned to the new boundary and RX_BYTE_IS_ALIGNED_O will switch to 1.
RX_MCOMMA_ALIGN_I	1	Input	Enable minus comma detection and alignment.
RX_PCOMMA_ALIGN_I	1	Input	Enable plus comma detection and alignment.
RX_SLIDE_I	1	Input	If automatic comma alignment is not used port can be used for manual comma alignment. A single cycle strobe slides the incoming data stream by one bit depending on RX_SLIDE_MODE setting. RX_SLIDE_I must be deasserted for 11 RX_CLK_O cycles before next slide operation can be requested. RX_SLIDE_I must be synchron with RX_CLK_O.
RX_COMMA_DETECT_EN_I	1	Input	Enable or bypass comma detection and alignment circuit.

To facilitate this alignment, transmitters embed a recognizable sequence known as a comma into the data stream. This comma acts as a delimiter, marking specific points in the data that the receiver can use to establish alignment. Upon receiving the data, the receiver searches for this comma, which helps in identifying byte boundaries and aligning data words accurately.

Table 2.50: *Receiver byte and word alignment configuration in the register file*

Register File

Name	Width	Mode	Default	Description
RX_ALIGN_MCOMMA_VALUE	10	r/w	0x283	
RX_MCOMMA_ALIGN_OVR	1	r/w	0	Receiver negative comma alignment enable overwrite when SerDes in testmode.
RX_MCOMMA_ALIGN	1	r/w	0	Receiver negative comma alignment enable value when RX_MCOMMA_ALIGN_OVR is active.
RX_ALIGN_PCOMMA_VALUE	10	r/w	0x17C	
RX_PCOMMA_ALIGN_OVR	1	r/w	0	Receiver positive comma alignment enable overwrite when SerDes in testmode.
RX_PCOMMA_ALIGN	1	r/w	0	Receiver positive comma alignment enable value when RX_PCOMMA_ALIGN_OVR is active.
RX_ALIGN_COMMA_WORD	2	r/w	0	
RX_ALIGN_COMMA_ENABLE	10	r/w	0x3FF	Mask for RX_ALIGN_MCOMMA_VALUE and RX_ALIGN_PCOMMA_VALUE.
RX_SLIDE_MODE	2	r/w	0	Bit 0: RX_SLIDE_BIT Bit 1: RX_SLIDE_BYTE
RX_COMMA_DETECT_EN_OVR	1	r/w	0	Receiver comma detection overwrite when SerDes is in testmode.
RX_COMMA_DETECT_EN	1	r/w	0	Receiver comma detection enable when RX_COMMA_DETECT_EN_OVR is active.
RX_SLIDE	2	r/w	0	
RX_BYTE_IS_ALIGNED	1	r	0	High if comma detection has aligned incoming commas to the byte boundaries successfully.
RX_BYTE_REALIGN	1	r	0	High if alignment circuit has detected an unaligned comma and realigned it to the new boundary.

By moving the detected comma to the correct byte boundary, the receiver ensures that the parallel data matches the transmitted format. This process is essential for maintaining data integrity and synchronizing received data with its intended structure, thereby minimizing errors and ensuring reliable data communication.

2.5.6.6 Receiver Pattern Checker

The pattern checker within the receiver segment of the SerDes employs the industry-standard pseudo-random bit sequences (PRBS) PRBS-7, PRBS-15, PRBS-23 and PRBS-31 to detect and validate the received data patterns. It is self-synchronizing and indicates it's lock state with the RX_PRBS_LOCKED parameter. RX_PRBS_ERR_CNT indicates single bit errors, and is cleared if the following incoming data is correct. The RX_PRBS_ERR_CNT counter is incremented in the event of an error. The error counter stops when reaching 0x 7F FF and can be is set back to zero when RX_PRBS_CNT_RESET_I or RX_PCS_RESET_I is active.

Table 2.51: *CC_SERDES interface for the pattern checker*

Name	Width	Direction	Description
RX_PRBS_SEL_I	3	Input	Receiver PRBS mode selection 000: Bypass PRBS generator 001: PRBS-7 010: PRBS-15 011: PRBS-23 100: PRBS-31 101: Reserved 110: Reserved 111: Reserved
RX_PRBS_CNT_RESET_I	1	Input	Reset the PRBS checker error counter
RX_PRBS_ERR_O	1	Output	Reset the PRBS checker error counter

Table 2.52: *Pattern checker configuration in the register file*

Register File

Name	Width	Mode	Default	Description
RX_PRBS_OVR	1	r/w	0	PRBS overwrite when SerDes is in test-mode
RX_PRBS_SEL	3	r/w	0	Receiver PRBS mode selection when RX_PRBS_OVR is active
RX_PRBS_CNT_RESET	3	r/w	0	Asynchronous PRBS reset

Table 2.53: *Pattern checker status in the register file*

Register File

Name	Width	Mode	Default	Description
RX_PRBS_ERR_CNT	15	r	0	PRBS error counter. RX_PRBS_CNT_RESET resets the counter.
RX_PRBS_LOCKED	1	r	0	

2.5.6.7 RX 8B/10B Decoder

The SerDes has a build-in 8B / 10B decoder. 8b / 10b encoding / decoding is a widely-used technique in high-speed data transmission to ensure reliable and efficient communication. In decoding 8b / 10b-encoded data, the 10-bit code is converted back into its original 8-bit form by the receiver. This process involves error detection and correction using the added bits, restoring the original data from the transmitted signal. By reversing the encoding process, the receiver retrieves the accurate digital information, ensuring the integrity and reliability of the transmitted data in high-speed communication systems.

Table 2.54: *CC_SERDES interface for the 8B / 10B decoder*

Name	Width	Direction	Description
RX_8B10B_BYPASS_I	8	Input	Each high bit bypasses corresponding byte
RX_8B10B_EN_I	1	Input	Enable or disable 8B / 10B decoder
RX_CHAR_IS_COMMA_O	8	Output	Bit is high if corresponding RX_DATA_I byte is a valid comma character
RX_CHAR_IS_K_O	8	Output	High if corresponding RX_DATA_I byte is a valid K character
RX_DISP_ERR_O	8	Output	High for corresponding RX_DATA_I byte disparity error
RX_NOT_IN_TABLE_O	8	Output	High if corresponding RX_DATA_I byte is no valid character

Enabling the 8B / 10B decoder increases latency through the receiver path. However, it is possible to disable and bypass the 8B / 10B decoder to minimize latency, if it is not needed.

2.5.6.8 RX Elastic Buffer

The receiver elastic buffer is designed to bridge between two clock domains and features a clock correction mechanism. The elastic buffer may be bypassed if it is not used to reduce lantecy through the receiver data path.

Table 2.55: *CC_SERDES receiver elastic buffer interface*

Name	Width	Direction	Description
RX_BUF_RESET_I	1	Input	Resets and initializes the receiver's elastic buffer
RX_BUF_ERR_O	1	Output	High for buffer overflow or underflow, cleared by resetting elastic buffer

Table 2.56: *Elastic buffer configuration in the register file*

Register File

Name	Width	Mode	Default	Description
RX_BUF_RESET_TIME	5	r/w	3	Number of CLK_CORE_RX_I clock prescaled with RX_RESET_TIMER_PRESC cycles for that internal elastic buffer reset should be active
RX_BUF_RESET_OVR	1	r/w	0	RX elastic buffer reset overwrite when SerDes is in testmode
RX_BUF_RESET	1	r/w	0	RX elastic buffer reset value when RX_BUF_RESET_OVR is active
RX_CLKCOR_USE	1	r/w	0	0: Buffer used for phase alignment 1: Clock correction enabled
RX_CLKCOR_MIN_LAT	6	r/w	32	Specifies minimum elastic buffer latency. If buffer drops value, then circuit replicates incoming clock correction sequences to prevent underflow
RX_CLKCOR_MAX_LAT	6	r/w	39	Specifies maximum elastic buffer latency. If buffer exceeds value, then circuit removes incoming clock correction sequences to prevent overflow
RX_CLKCOR_SEQ_1_0	10	r/w	0x1F7	Clock correction sequence byte 0
RX_CLKCOR_SEQ_1_1	10	r/w	0x1F7	Clock correction sequence byte 1
RX_CLKCOR_SEQ_1_2	10	r/w	0x1F7	Clock correction sequence byte 2
RX_CLKCOR_SEQ_1_3	10	r/w	0x1F7	Clock correction sequence byte 3

2.5.7 Register file interface

The register file has a processor-friendly interface. It can be accessed via the FPGA fabric as well as the JTAG interface with priority given to the JTAG interface if access is simultaneous. Please see Section 3.4.7 on page 153 for the JTAG instructions.

Table 2.57: *CC_SERDES register file port description*

Port	Width	Direction	Description
REGFILE_CLK_I	1	Input	Register file interface clock
REGFILE_WE_I	1	Input	Register file write enable signal
REGFILE_EN_I	1	Input	Register file enable signal
REGFILE_ADDR_I	8	Input	Register file address bus
REGFILE_DI_I	16	Input	Register file data input bus
REGFILE_MASK_I	16	Input	Register file data input bus mask
REGFILE_DO_O	16	Output	Register file data output bus
REGFILE_RDY_O	1	Output	Register file ready indicator signal

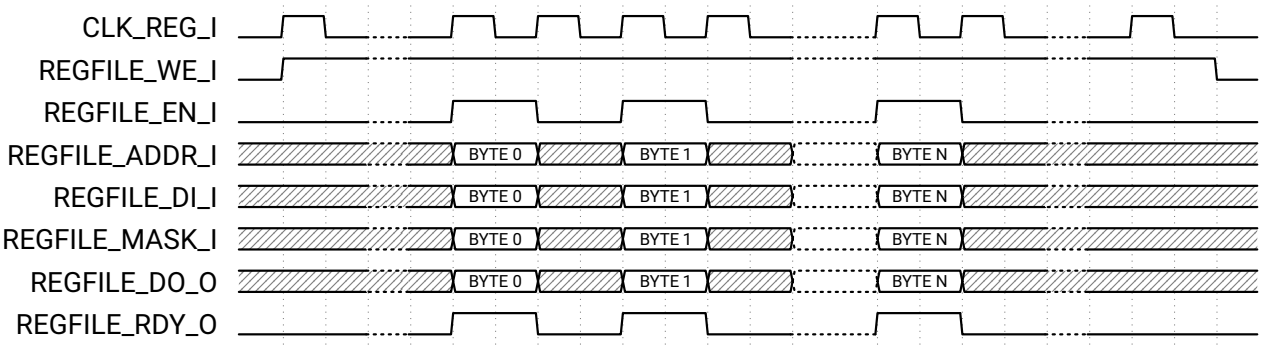


Figure 2.41: *Register file access from CPE array*

To write data to the register file from the core, REGFILE_WE_I and REGFILE_EN_I must be active high. The 16-bit data value at REGFILE_DI_I is written to the address of REGFILE_ADDR_I, with REGFILE_MASK_I being used as write-enable mask. Data is written to the register file with the rising edge of CLK_REG_I. At least 16 clock cycles must elapse between two accesses. The output REGFILE_RDY_O indicates the register file's status, as shown in Figure 2.41.

Table 2.58: *SerDes register file*

Address	I/O	Range	Default	Name
0x00	r/w	[4:0]	3	RX_BUF_RESET_TIME
	r/w	[9:5]	3	RX_PCS_RESET_TIME
	r/w	[14:10]	0	RX_RESET_TIMER_PRESC
	r/w	[15]	0	RX_RESET_DONE_GATE
0x01	r/w	[4:0]	3	RX_CDR_RESET_TIME
	r/w	[9:5]	3	RX_EQA_RESET_TIME
	r/w	[14:10]	3	RX_PMA_RESET_TIME
	r/w	[15]	1	RX_WAIT_CDR_LOCK
0x02	w/c	[0]	0	RX_CALIB_EN
	r	[1]	1	RX_CALIB_DONE
	r/w	[2]	0	RX_CALIB_OVR
	r/w	[6:3]	0	RX_CALIB_VAL
	r	[10:7]	0	RX_CALIB_CAL
	r/w	[13:11]	4	RX_RTERM_VCMSEL
	r/w	[14]	0	RX_RTERM_PD
	r	[15]	0	Unused
0x03	r/w	[7:0]	0xA3	RX_EQA_CKP_LF
	r/w	[15:8]	0xA3	RX_EQA_CKP_HF
0x04	r/w	[7:0]	0x01	RX_EQA_CKP_OFFSET
	r/w	[8]	0	RX_EN_EQA
	r/w	[12:9]	0	RX_EQA_LOCK_CFG
	r	[13]	0	RX_EQA_LOCKED
	r	[15:14]	0	Unused
0x05	r/w	[4:0]	8	RX_TH_MON1
	r/w	[5]	0	RX_EN_EQA_EXT_VALUE[0]
	r/w	[10:6]	8	RX_TH_MON2
	r/w	[11]	0	RX_EN_EQA_EXT_VALUE[1]
	r	[15:12]	0	Unused
0x06	r/w	[4:0]	8	RX_TAPW
	r/w	[5]	0	RX_EN_EQA_EXT_VALUE[2]
	r/w	[10:6]	8	RX_AFE_OFFSET
	r/w	[11]	0	RX_EN_EQA_EXT_VALUE[3]
	r	[15:12]	0	Unused
0x07	r	[4:0]	0	RX_EQA_TAPW
	r	[9:5]	0	RX_TH_MON
	r	[13:10]	0	RX_OFFSET
	r	[15:14]	0	Unused
0x08	r/w	[15:0]	0x01C0	RX_EQA_CONFIG
0x09	r/w	[4:0]	16	RX_AFE_PEAK
	r/w	[8:5]	8	RX_AFE_GAIN

Continued on next page

Table 2.58: *SerDes register file*

Continued from previous page

Address	I / O	Range	Default	Name
	r / w	[11:9]	4	RX_AFE_VCMSEL
	r	[15:12]	0	Unused
0x 0A	r / w	[7:0]	0x F8	RX_CDR_CKP
	r / w	[15:8]	0x 00	RX_CDR_CKI
0x 0B	r / w	[8:0]	128	RX_CDR_TRANS_TH
	r / w	[14:9]	0x 0B	RX_CDR_LOCK_CFG
	r	[15]	0	RX_CDR_LOCKED
0x 0C	r	[14:0]	0	RX_CDR_FREQ_ACC_VAL
	r	[15]	0	Unused
0x 0D	r	[15:0]	0	RX_CDR_PHASE_ACC_VAL
0x 0E	r / w	[14:0]	0	RX_CDR_FREQ_ACC
	r	[15]	0	Unused
0x 0F	r / w	[15:0]	0	RX_CDR_PHASE_ACC
0x 10	r / w	[1:0]	0	RX_CDR_SET_ACC_CONFIG
	r / w	[2]	0	RX_CDR_FORCE_LOCK
	r	[15:3]	0	Unused
0x 11	r / w	[9:0]	0x 2 83	RX_ALIGN_MCOMMA_VALUE
	r / w	[10]	0	RX_MCOMMA_ALIGN_OVR
	r / w	[11]	0	RX_MCOMMA_ALIGN
	r	[15:12]	0	Unused
0x 12	r / w	[9:0]	0x 1 7C	RX_ALIGN_PCOMMA_VALUE
	r / w	[10]	0	RX_PCOMMA_ALIGN_OVR
	r / w	[11]	0	RX_PCOMMA_ALIGN
	r / w	[13:12]	0	RX_ALIGN_COMMA_WORD
	r	[15:14]	0	Unused
0x 13	r / w	[9:0]	0x 3 FF	RX_ALIGN_COMMA_ENABLE
	r / w	[11:10]	0	RX_SLIDE_MODE
	r / w	[12]	0	RX_COMMA_DETECT_EN_OVR
	r / w	[13]	0	RX_COMMA_DETECT_EN
	r / w	[14]	0	RX_SLIDE[0]
	w / c	[15]	0	RX_SLIDE[1]
0x 14	w / c	[0]	0	RX_EYE_MEAS_EN
	r / w	[15:1]	0	RX_EYE_MEAS_CFG
0x 15	r / w	[5:0]	0	RX_MON_PH_OFFSET
	r	[15:6]	0	Unused
0x 16	r	[15:0]	0	RX_EYE_MEAS_CORRECT_11S

Continued on next page

Table 2.58: *SerDes register file*

Continued from previous page

Address	I / O	Range	Default	Name
0x17	r	[15:0]	0	RX_EYE_MEAS_WRONG_11S
0x18	r	[15:0]	0	RX_EYE_MEAS_CORRECT_00S
0x19	r	[15:0]	0	RX_EYE_MEAS_WRONG_00S
0x1A	r	[15:0]	0	RX_EYE_MEAS_CORRECT_001S
0x1B	r	[15:0]	0	RX_EYE_MEAS_WRONG_001S
0x1C	r	[15:0]	0	RX_EYE_MEAS_CORRECT_110S
0x1D	r	[15:0]	0	RX_EYE_MEAS_WRONG_110S
0x1E	r / w	[3:0]	4	RX_EI_BIAS
	r / w	[7:4]	4	RX_EI_BW_SEL
	r / w	[8]	0	RX_EN_EI_DETECTOR_OVR
	r / w	[9]	0	RX_EN_EI_DETECTOR
	r	[10]	0	RX_EI_EN
	r	[15:11]	0	Unused
0x1F	r	[14:0]	0	RX_PRBS_ERR_CNT
	r	[15]	0	RX_PRBS_LOCKED
0x20	r / w	[0]	0	Read: RX_DATA[0] Write: RX_DATA_SEL
	r	[15:1]	0	RX_DATA[15:1]
0x21	r	[15:0]	0	RX_DATA[31:16]
0x22	r	[15:0]	0	RX_DATA[47:32]
0x23	r	[15:0]	0	RX_DATA[63:48]
0x24	r	[15:0]	0	RX_DATA[79:64]
0x25	r / w	[0]	0	RX_BUF_BYPASS
	r / w	[1]	0	RX_CLKCOR_USE
	r / w	[7:2]	32	RX_CLKCOR_MIN_LAT
	r / w	[13:8]	39	RX_CLKCOR_MAX_LAT
	r	[15:14]	0	Unused
0x26	r / w	[9:0]	0x1F7	RX_CLKCOR_SEQ_1_0
	r	[15:10]	0	Unused
0x27	r / w	[9:0]	0x1F7	RX_CLKCOR_SEQ_1_1
	r	[15:10]	0	Unused
0x28	r / w	[9:0]	0x1F7	RX_CLKCOR_SEQ_1_2

Continued on next page

Table 2.58: *SerDes register file*

Continued from previous page

Address	I/O	Range	Default	Name
	r	[15:10]	0	Unused
0x29	r/w	[9:0]	0x1F7	RX_CLKCOR_SEQ_1_3
	r	[15:10]	0	Unused
0x2A	r/w	[0]	0	RX_PMA_LOOPBACK
	r/w	[1]	0	RX_PCS_LOOPBACK
	r/w	[3:2]	3	RX_DATAPATH_SEL
	r/w	[4]	0	RX_PRBS_OVR
	r/w	[7:5]	0	RX_PRBS_SEL
	r/w	[8]	0	RX_LOOPBACK_OVR
	w/c	[9]	0	RX_PRBS_CNT_RESET
	r/w	[10]	0	RX_POWER_DOWN_OVR
	r/w	[11]	0	RX_POWER_DOWN_N
	r	[12]	0	RX_PRESENT
	r	[13]	0	RX_DETECT_DONE
	r	[14]	0	RX_BUF_ERR
	r	[15]	0	Unused
0x2B	r/w	[0]	0	RX_RESET_OVR
	w/c	[1]	0	RX_RESET
	r/w	[2]	0	RX_PMA_RESET_OVR
	w/c	[3]	0	RX_PMA_RESET
	r/w	[4]	0	RX_EQA_RESET_OVR
	w/c	[5]	0	RX_EQA_RESET
	r/w	[6]	0	RX_CDR_RESET_OVR
	w/c	[7]	0	RX_CDR_RESET
	r/w	[8]	0	RX_PCS_RESET_OVR
	w/c	[9]	0	RX_PCS_RESET
	r/w	[10]	0	RX_BUF_RESET_OVR
	w/c	[11]	0	RX_BUF_RESET
	r/w	[12]	0	RX_POLARITY_OVR
	r/w	[13]	0	RX_POLARITY
	r/w	[14]	0	RX_8B10B_EN_OVR
	r/w	[15]	0	RX_8B10B_EN
0x2C	r/w	[7:0]	0	RX_8B10B_BYPASS
	r	[8]	0	RX_BYTE_IS_ALIGNED
	r/c	[9]	0	RX_BYTE_REALIGN
	r	[10]	0	RX_RESET_DONE
	r	[15:11]	0	Unused
0x2D	w/c	[0]	0	RX_DBG_EN
	r/w	[4:1]	0	RX_DBG_SEL
	r/w	[5]	0	RX_DBG_MODE
	r/w	[11:6]	0x05	RX_DBG_SRAM_DELAY
	r	[15:12]	0	Unused

Continued on next page

Table 2.58: *SerDes register file*

Continued from previous page

Address	I/O	Range	Default	Name
0x2E	r/w	[9:0]	0	RX_DBG_ADDR
	w/c	[10]	0	RX_DBG_RE
	w/c	[11]	0	RX_DBG_WE
	r/w	[15:12]	0	RX_DBG_DATA[3:0]
0x2F	r/w	[15:0]	0	RX_DBG_DATA[19:4]
0x30	r/w	[4:0]	0	TX_SEL_PRE
	r/w	[9:5]	0	TX_SEL_POST
	r/w	[14:10]	15	TX_AMP
	r	[15]	0	Unused
0x31	r/w	[4:0]	0	TX_BRANCH_EN_PRE
	r/w	[10:5]	0x3F	TX_BRANCH_EN_MAIN
	r/w	[15:11]	0	TX_BRANCH_EN_POST
0x32	r/w	[2:0]	4	TX_TAIL_CASCODE
	r/w	[9:3]	63	TX_DC_ENABLE
	r/w	[14:10]	0	TX_DC_OFFSET
	r	[15]	0	Unused
0x33	r/w	[4:0]	0	TX_CM_RAISE
	r/w	[9:5]	14	TX_CM_THRESHOLD_0
	r/w	[14:10]	16	TX_CM_THRESHOLD_1
	r	[15]	0	Unused
0x34	r/w	[4:0]	0	TX_SEL_PRE_EI
	r/w	[9:5]	0	TX_SEL_POST_EI
	r/w	[14:10]	15	TX_AMP_EI
	r	[15]	0	Unused
0x35	r/w	[4:0]	0	TX_BRANCH_EN_PRE_EI
	r/w	[10:5]	0x3F	TX_BRANCH_EN_MAIN_EI
	r/w	[15:11]	0	TX_BRANCH_EN_POST_EI
0x36	r/w	[2:0]	4	TX_TAIL_CASCODE_EI
	r/w	[9:3]	63	TX_DC_ENABLE_EI
	r/w	[14:10]	0	TX_DC_OFFSET_EI
	r	[15]	0	Unused
0x37	r/w	[4:0]	0	TX_CM_RAISE_EI
	r/w	[9:5]	14	TX_CM_THRESHOLD_0_EI
	r/w	[14:10]	16	TX_CM_THRESHOLD_1_EI
	r	[15]	0	Unused
0x38	r/w	[4:0]	0	TX_SEL_PRE_RXDET
	r/w	[9:5]	0	TX_SEL_POST_RXDET
	r/w	[14:10]	15	TX_AMP_RXDET

Continued on next page

Table 2.58: *SerDes register file*

Continued from previous page

Address	I/O	Range	Default	Name
	r	[15]	0	Unused
0x39	r/w	[4:0]	0	TX_BRANCH_EN_PRE_RXDET
	r/w	[10:5]	0x3F	TX_BRANCH_EN_MAIN_RXDET
	r/w	[15:11]	0	TX_BRANCH_EN_POST_RXDET
0x3A	r/w	[2:0]	4	TX_TAIL_CASCODE_RXDET
	r/w	[9:3]	63	TX_DC_ENABLE_RXDET
	r/w	[14:10]	0	TX_DC_OFFSET_RXDET
	r	[15]	0	Unused
0x3B	r/w	[4:0]	0	TX_CM_RAISE_RXDET
	r/w	[9:5]	14	TX_CM_THRESHOLD_0_RXDET
	r/w	[14:10]	16	TX_CM_THRESHOLD_1_RXDET
	r	[15]	0	Unused
0x3C	w/c	[0]	0	TX_CALIB_EN
	r	[1]	1	TX_CALIB_DONE
	r/w	[2]	0	TX_CALIB_OVR
	r/w	[6:3]	0	TX_CALIB_VAL
	r	[10:7]	0	TX_CALIB_CAL
	r	[15:11]	0	Unused
0x3D	r/w	[7:0]	0x80	TX_CM_REG_KI
	r/w	[8]	0	TX_CM_SAR_EN
	r/w	[9]	1	TX_CM_REG_EN
	r	[15:10]	0	Unused
0x3E	r	[4:0]	0	TX_CM_SAR_RESULT_0
	r	[9:5]	0	TX_CM_SAR_RESULT_1
	r	[15:10]	0	Unused
0x3F	r/w	[4:0]	3	TX_PMA_RESET_TIME
	r/w	[9:5]	3	TX_PCS_RESET_TIME
	r/w	[10]	0	TX_PCS_RESET_OVR
	w/c	[11]	0	TX_PCS_RESET
	r/w	[12]	0	TX_PMA_RESET_OVR
	w/c	[13]	0	TX_PMA_RESET
	r/w	[14]	0	TX_RESET_OVR
	w/c	[15]	0	TX_RESET
0x40	r/w	[1:0]	0	TX_PMA_LOOPBACK
	r/w	[2]	0	TX_PCS_LOOPBACK
	r/w	[4:3]	3	TX_DATAPATH_SEL
	r/w	[5]	0	TX_PRBS_OVR
	r/w	[8:6]	0	TX_PRBS_SEL
	w/c	[9]	0	TX_PRBS_FORCE_ERR
	r/w	[10]	0	TX_LOOPBACK_OVR

Continued on next page

Table 2.58: *SerDes register file*

Continued from previous page

Address	I/O	Range	Default	Name
	r/w	[11]	0	TX_POWER_DOWN_OVR
	r/w	[12]	0	TX_POWER_DOWN_N
	r	[15:13]	0	Unused
0x41	r/w	[0]	0	TX_ELEC_IDLE_OVR
	r/w	[1]	0	TX_ELEC_IDLE
	r/w	[2]	0	TX_DETECT_RX_OVR
	r/w	[3]	0	TX_DETECT_RX
	r/w	[4]	0	TX_POLARITY_OVR
	r/w	[5]	0	TX_POLARITY
	r/w	[6]	0	TX_8B10B_EN_OVR
	r/w	[7]	0	TX_8B10B_EN
	r/w	[8]	0	TX_DATA_OVR
	r/w	[11:9]	0	TX_DATA_CNT
	w/c	[12]	0	TX_DATA_VALID
	r	[13]	0	TX_BUF_ERR
	r	[14]	0	TX_RESET_DONE
	r	[15]	0	Unused
0x42	r/w	[15:0]	0	TX_DATA
0x50	r/w	[0]	0	PLL_EN_ADPLL_CTRL
	r/w	[1]	0	PLL_CONFIG_SEL
	r/w	[2]	0	PLL_SET_OP_LOCK
	r/w	[3]	0	PLL_ENFOCRE_LOCK
	r/w	[4]	0	PLL_DISABLE_LOCK
	r/w	[5]	1	PLL_LOCK_WINDOW
	r/w	[6]	1	PLL_FAST_LOCK
	r/w	[7]	0	PLL_SYNC_BYPASS
	r/w	[8]	0	PLL_PFD_SELECT
	r/w	[9]	0	PLL_REF_BYPASS
	r/w	[10]	0	PLL_REF_SEL
	r/w	[11]	1	PLL_REF_RTERM
	r	[15:12]	0	Unused
0x51	r/w	[5:0]	58	PLL_FCNTRL
	r/w	[11:6]	27	PLL_MAIN_DIVSEL
	r/w	[13:12]	0	PLL_OUT_DIVSEL
	r	[15:14]	0	Unused
0x52	r/w	[4:0]	3	PLL_CI
	r/w	[14:5]	80	PLL_CP
	r	[15]	0	Unused
0x53	r/w	[3:0]	0	PLL_AO
	r/w	[6:4]	0	PLL_SCAP
	r/w	[8:7]	2	PLL_FILTER_SHIFT
	r/w	[11:9]	2	PLL_SAR_LIMIT

Continued on next page

Table 2.58: *SerDes register file*

Continued from previous page

Address	I / O	Range	Default	Name
	r	[15:12]	0	Unused
0x 54	r / w	[10:0]	512	PLL_FT
	r / w	[11]	0	PLL_OPEN_LOOP
	r / w	[12]	1	PLL_SCAP_AUTO_CAL
	r	[15:13]	0	Unused
0x 55	r	[0]	0	PLL_LOCKED
	r	[1]	0	PLL_CAP_FT_OF
	r	[2]	0	PLL_CAP_FT_UF
	r	[12:3]	0	PLL_CAP_FT
	r	[14:13]	0	PLL_CAP_STATE
	r	[15]	0	Unused
0x 56	r	[7:0]	0	PLL_SYNC_VALUE
	r	[15:8]	0	Unused
0x 57	r / w	[2:0]	4	PLL_BISC_MODE
	r / w	[6:3]	15	PLL_BISC_TIMER_MAX
	r / w	[7]	0	PLL_BISC_OPT_DET_IND
	r / w	[8]	0	PLL_BISC_PFD_SEL
	r / w	[9]	0	PLL_BISC_DLY_DIR
	r / w	[12:10]	1	PLL_BISC_COR_DLY
	r / w	[13]	0	PLL_BISC_CAL_SIGN
	r / w	[14]	1	PLL_BISC_CAL_AUTO
	r	[15]	0	Unused
0x 58	r / w	[4:0]	4	PLL_BISC_CP_MIN
	r / w	[9:5]	18	PLL_BISC_CP_MAX
	r / w	[14:10]	12	PLL_BISC_CP_START
	r	[15]	0	Unused
0x 59	r / w	[4:0]	0	PLL_BISC_DLY_PFD_MON_REF
	r / w	[9:5]	2	PLL_BISC_DLY_PFD_MON_DIV
	r	[15:10]	0	Unused
0x 5A	r	[0]	0	PLL_BISC_TIMER_DONE
	r	[7:1]	0	PLL_BISC_CP
	r	[15:8]	0	Unused
0x 5B	r	[15:0]	0	PLL_BISC_CO
0x 5C	r / w	[0]	0	SerDes Enable
	r / w	[1]	0	SerDes Auto Init
	r / w	[2]	0	SerDes Testmode
	r	[15:3]	0	Unused

2.6 Clock Generators (PLLs)

2.6.1 Overview

In each FPGA die four independent PLL clock generators are available, i.e. there are 4 PLLs in CCGM1A1 and 8 PLLs in CCGM1A2. They have the following properties:

- ADPLL clock generator for versatile FPGA clock generation.
- Clock generation based on a digitally controlled oscillator (DCO) running with a typical frequency tuning range from 1 GHz to 2 GHz.
- Programmable clock frequency dividers for reference clock input, PLL loop divider and clock output divider, enabling wide output and reference frequency ranges.
- Clock outputs available in four 90°-spaced phases.
- Fast lock-in by binary frequency search.
- Autonomous free-running oscillator mode for FPGA configuration clock after power-up.

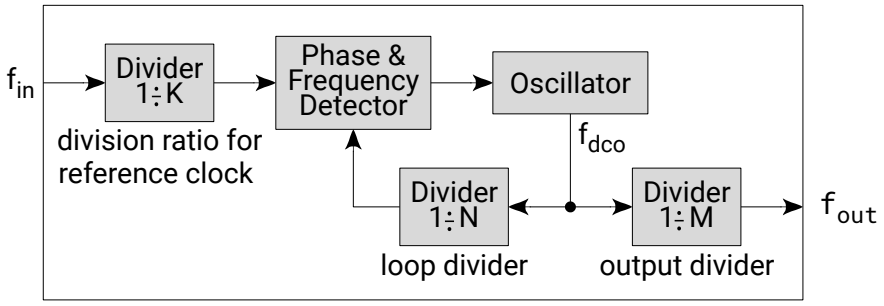


Figure 2.42: *General PLL structure*

Figure 2.42 shows the general PLL structure. The reference clock f_{in} can be chosen individually for each PLL:

- Clock input `CLK0` (2nd function of pin `N14`, 1st function is GPIO `IO_SB_A8`)
- Clock input `CLK1` (2nd function of pin `P12`, 1st function is GPIO `IO_SB_A7`)
- Clock input `CLK2` (2nd function of pin `P14`, 1st function is GPIO `IO_SB_A6`)
- Clock input `CLK3` (2nd function of pin `R13`, 1st function is GPIO `IO_SB_A5`)
- SerDes clock input `SER_CLK` (pin `T12`, can also be used as general purpose clock input)
- `SER_CLK` can be used both in single-ended mode and together with `SER_CLK_N` (pin `T13`) in LVDS mode

Reference clocks can be used simultaneously by several PLLs.

The allowed oscillator frequency f_{dco} depends on the FPGA supply voltage as follows:

Low power mode:	$(VDD_{PLL} = 0.9\,V \pm 50\,mV)$:	$500\,MHz \leqslant f_{dco} \leqslant 1{,}000\,MHz$
Economy mode:	$(VDD_{PLL} = 1.0\,V \pm 50\,mV)$:	$1{,}000\,MHz \leqslant f_{dco} \leqslant 2{,}000\,MHz$
Speed mode:	$(VDD_{PLL} = 1.1\,V \pm 50\,mV)$:	$1{,}250\,MHz \leqslant f_{dco} \leqslant 2{,}500\,MHz$

2.6.2 Clock signals of the CCGM1A2 chip

The BGA pins T12 (SER_CLK) and T13 (SER_CLK_N) of CCGM1A2 connects the SerDes clock inputs of both FPGA dies.

Likewise, the four dedicated clock input pins N14 (CLK0), P12 (CLK1), P14 (CLK2) and R13 (CLK3) are also each internally connected to both FPGA dies.

2.6.3 PLL core

The PLL core module has one clock input for the reference clock and four clock outputs, since the output clock is available in all 90°-spaced phases as shown in Figure 2.43. An additional clock input PLL_FB can be used for an external feedback path. Table 2.59 gives an overview of the PLL core ports.

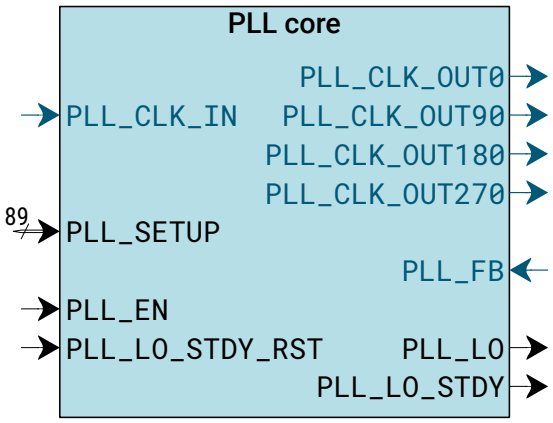


Figure 2.43: *Ports of the PLL core*

The simplified block diagram in Figure 2.44 shows that some bits of the setup vector PLL_SETUP are used for setting the five programmable dividers. Other bits of the vector are used to switch on an additional divider by 2 in the signal path (p_{osc}) or to change the signal path itself (p_{loop}, p_{out}). This results in different divider ratios for the loop and the output path as shown in Table 2.61.

The programmable dividers can be used in the following ranges:

$1 \leqslant K \leqslant 4095$ (12 bits)

$1 \leqslant N_1 \leqslant 63$	(6 bits)	$1 \leqslant M_1 \leqslant 63$	(6 bits)
$1 \leqslant N_2 \leqslant 1023$	(10 bits)	$1 \leqslant M_2 \leqslant 1023$	(10 bits)

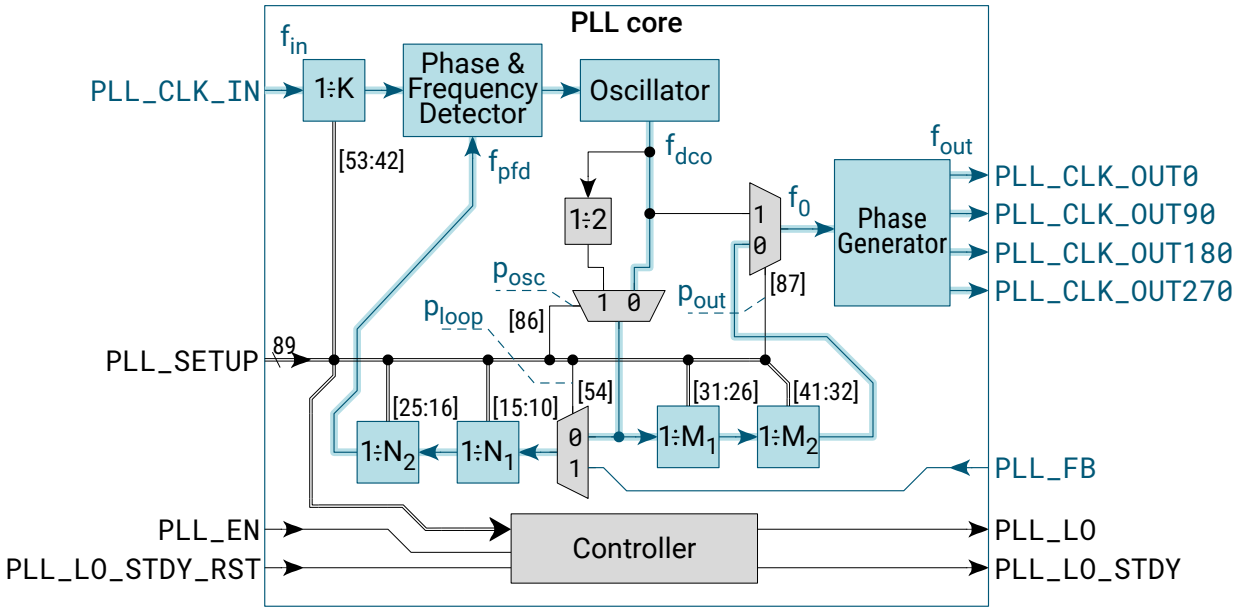


Figure 2.44: *Block diagram of the PLL core*

Table 2.59: *Ports of the PLL core*

Name	Width	Direction	Description
PLL_CLK_IN	1	input	Reference clock
PLL_CLK_OUT0	1	output	PLL output, 0° phase
PLL_CLK_OUT90	1	output	PLL output, 90° phase
PLL_CLK_OUT180	1	output	PLL output, 180° phase
PLL_CLK_OUT270	1	output	PLL output, 270° phase
PLL_FB	1	input	PLL feedback for external loop path
PLL_SETUP	1	input	Setup selection vector
PLL_EN	1	input	PLL enable signal
PLL_LO	1	output	PLL locked signal
PLL_LO_STDY	1	output	Steady PLL locked signal
PLL_LO_STDY_RST	1	input	Reset for PLL_LO_STEADY

The maximum input frequencies of the PLL dividers is shown in Table 2.60. This must be taken into account when a desired divider value is split between the two dividers connected in series.

With the parameter value p_{loop} = PLL_SETUP[54] = 0 (selection of the feedback path) the loop is completely inside the PLL module and the oscillator frequency is

$$f_{dco} = f_{in} \frac{N_1 \cdot N_2 \cdot (p_{osc} + 1)}{K}$$

Table 2.60: *Maximum input frequency of the PLL dividers*

Divider	Low power	Economy	High speed	Description
N1	1.000 GHz	1.250 GHz	1.666 GHz	High-speed part of the loop path divider
N2	0.500 GHz	0.6125 GHz	0.833 GHz	Low-speed part of the loop path divider
M1	1.000 GHz	1.250 GHz	1.666 GHz	High-speed part of the output path divider
M2	0.500 GHz	0.6125 GHz	0.833 GHz	Low-speed part of the output path divider

where parameter p_{osc} = PLL_SETUP[86] can take the values $\{0, 1\}$.

If in the other case $p_{loop} = 1$, then the loop is routed via the PLL output through the user circuit and back to the PLL. The oscillator frequency is given by

$$f_{dco} = f_{in} \frac{2 \cdot N_1 \cdot N_2}{K} \cdot \begin{cases} M_1 \cdot M_2 \cdot (p_{osc} + 1), & p_{out} = 0 \\ 1, & p_{out} = 1 \end{cases} .$$

The PLL output clock depends on the parameters p_{out} and p_{osc}. It is

$$f_{out} = \frac{f_{dco}}{2} \cdot \begin{cases} \frac{1}{M_1 \cdot M_2 \cdot (p_{osc}+1)}, & p_{out} = 0 \\ 1, & p_{out} = 1 \end{cases}$$

and after inserting f_{dco}

Table 2.61: *Divider ratio of loop and output path*

PLL_SETUP				
[54]	[86]	[87]	Loop path	Output path
p_{loop}	p_{osc}	p_{out}	$f_{dco} \rightarrow f_{pfd}$	$f_{dco} \rightarrow f_{out}$
0	0	0	$N_1 \cdot N_2$	$M_1 \cdot M_2 \cdot 2$
0	0	1	$N_1 \cdot N_2$	2
0	1	0	$2 \cdot N_1 \cdot N_2$	$2 \cdot M_1 \cdot M_2 \cdot 2$
0	1	1	$2 \cdot N_1 \cdot N_2$	2
1	0	0	$M_1 \cdot M_2 \cdot 2 \cdot N_1 \cdot N_2$	$M_1 \cdot M_2 \cdot 2$
1	0	1	$2 \cdot N_1 \cdot N_2$	2
1	1	0	$2 \cdot M_1 \cdot M_2 \cdot 2 \cdot N_1 \cdot N_2$	$2 \cdot M_1 \cdot M_2 \cdot 2$
1	1	1	$2 \cdot N_1 \cdot N_2$	2

$$f_{out} = f_{in} \frac{N_1 \cdot N_2}{2 \cdot K} \cdot \begin{cases} \frac{1}{M_1 \cdot M_2}, & p_{out} = 0 \\ (p_{osc} + 1), & p_{out} = 1 \end{cases}$$

for the internal feedback path with $p_{loop} = 0$ and

$$f_{out} = f_{in} \frac{N_1 \cdot N_2}{K}$$

for the external feedback path with $p_{loop} = 1$.

Duty cycle and phase

All 90° phase shiftings can only be ensured if one of the two conditions is met:

1. $M_1 = M_2 = 1$
2. M_2 is even and $M_2 > 0$

If these conditions are not fulfilled, the input signal f_0 of the phase generator will not have 50% duty cycle and therefore the phases are not 90°for all signals.

Definitions:		
	a	Pulse width of f_0
	b	Space width of f_0
	$T_0 = a + b$	Period of f_0
	$D_0 = \frac{a}{T_0}$	Duty cycle of f_0

The DCO output clock f_{dco} has always 50% duty cycle. Behind the output dividers, the duty cycle $D_0 = D(f_0)$ depends on the following conditions:

a)	$M_1 = M_2 = 1$:	No dividers are involved, which is why $f_0 = f_{dco}$ and $D_0 = 50\%$.
b)	$M_1 > 1$ and $M_2 = 1$:	The pulse width of f_0 is the same as for f_{dco}. This leads to a duty cycle $D_0 = \frac{1}{2M_1}$ which is poor for all M_1 ($0 < D_0 \leqslant 1/4$).
c)	M_2 is even and $M_2 > 0$:	$D_0 = 50\%$.
d)	M_2 is odd and $M_2 \geqslant 3$:	$D_0 = \frac{1}{2} - \frac{1}{2M_2}$ which is near to 50% for large M_2 ($1/3 \leqslant D_0 < 1/2$).

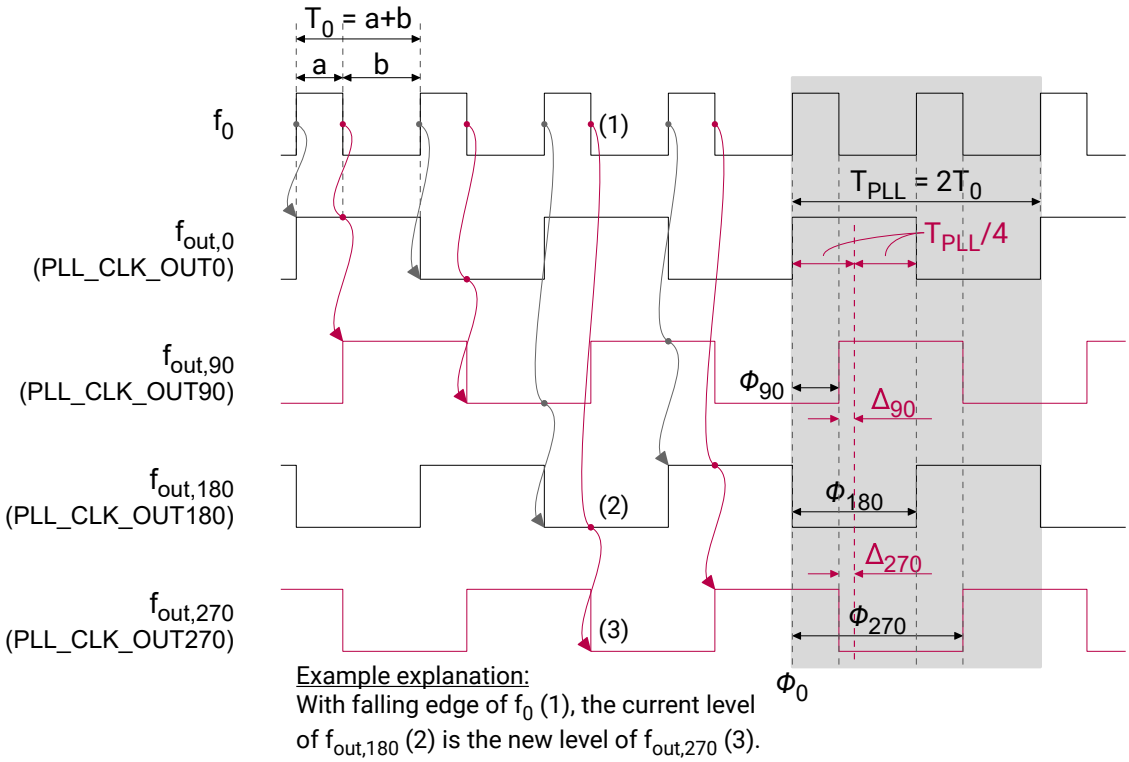


Figure 2.45: *Output signals f_{out} of the phase generator*

Definitions:

$T_{PLL} = 2T_0$	Period of the phase generator output signals f_{out} (all 4 phases)
$D_0 = \frac{a}{T_0} = \frac{2a}{T_{PLL}}$	Duty cycle of f_0
$\phi_0 = 0°$	Reference phase of f_0
$\phi(f_{out,0°}) = \phi_0 = 0°$	Phase of `PLL_CLK_OUT0`
$\phi(f_{out,90°}) = \phi_{90}$	Phase of `PLL_CLK_OUT90`, depends on D_0
$\Delta_{90} = \phi_{90} - 90°$	Phase deviation of $f_{out,90°}$
$\phi(f_{out,180°}) = \phi_{180} = 180°$	Phase of `PLL_CLK_OUT180`
$\phi(f_{out,270°}) = \phi_{270}$	Phase of `PLL_CLK_OUT270`, depends on D_0
$\Delta_{270} = \phi_{270} - 270°$	Phase deviation of $f_{out,270°}$

Figure 2.45 shows the input clock f_0 of the phase generator and the four output clocks f_{out}. The duty cycle is in any case $D_0 \leqslant 50\%$ which leads to $\Delta \leqslant 0$. The phase generator always generates a duty cycle of 50% regardless of D_0. Table 2.62 shows the formulas used to calculate the characteristics of the output signals f_{out}. The input clock f_0 specifies the origin $\phi_0 = 0°$ for the output signals.

Table 2.62: *Phase generator output characteristics*

Signal	Characteristic	Formula
$f_{out,0}$	phase	$\phi_0 = 0°$
	phase error	$\Delta_0 = 0°$
$f_{out,90}$	phase	$\phi_{90} = \frac{a}{T_{PLL}} \cdot 360° = 2D_0 \cdot 90°$
	phase error	$\Delta_{90} = \phi_{90} - 90° = (2D_0 - 1) \cdot 90°$
$f_{out,180}$	phase	$\phi_{180} = \frac{T_0}{T_{PLL}} \cdot 360° = \frac{T_{PLL}/2}{T_{PLL}} \cdot 360° = 180°$
	phase error	$\Delta_{180} = \phi_{180} - 180° = 0°$
$f_{out,270}$	phase	$\phi_{270} = \frac{2a+b}{T_{PLL}} \cdot 360° = 180° + 2D_0 \cdot 90°$
	phase error	$\Delta_{270} = \phi_{270} - 270° = 180° + 2D_0 \cdot 90° - 270° = (2D_0 - 1) \cdot 90° = \Delta_{90}$

Lock signals

Figure 2.46 shows how the locking mechanism works:

(A) When the PLL reaches its locked state, this is indicated by a rising edge at PLL_LO. If this is the first rising edge of PLL_LO, signal PLL_LO_STDY gets a rising edge as well.

(B) If the PLL loses the locked state, PLL_LO as well as PLL_LO_STDY fall back to low level.

(C) Reaching locked state again, indicated by a rising edge of PLL_LO, signal PLL_LO_STDY stays low to indicate that the locked state was not stable all the time.

(D) Only when PLL_LO_STDY_RST gets a pulse of at least 2 clock cycles of PLL_CLK_IN (f_{in}), ...

(E) ...the next rising edge of PLL_LO also leads again to the high level of PLL_LO_STDY.

Summarized, PLL_LO_STDY indicates that the locked state was reached and then remained stable permanently.

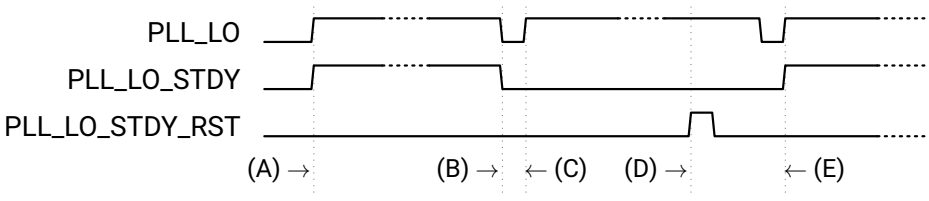


Figure 2.46: *Timing diagram of the locking mechanism*

Autonomous Mode

The ADPLL provides an autonomous mode for configuration of the CCGM1A1 / CCGM1A2 where the ADPLL can generate an output clock without a reference clock. In this case the ADPLL is in open loop mode and the provided frequency is smaller than 50 MHz to ensure save operation with most flash memories (for details see below on this page). The frequency range depends on the power supply voltage and temperature as shown in Table 2.63.

In autonomous mode, the ADPLL does not require any configurations, such as divider settings or lock-in configurations. The configuration vector PLL_SETUP is ignored in the autonomous mode.

Table 2.63: *ADPLL output frequencies in autonomous mode*

VDD_PLL	Power mode	min (-40°)	max (+125°)
0.85 V	Low Power	8.3 MHz	23.1 MHz
0.95 V	Low Power	15.0 MHz	31.4 MHz
0.95 V	Economy	15.0 MHz	31.4 MHz
1.05 V	Economy	22.9 MHz	39.9 MHz
1.05 V	High Speed	22.9 MHz	39.9 MHz
1.15 V	High Speed	31.8 MHz	49.0 MHz

If the GateMate FPGA should load the configuration automatically from a flash after reset, then the SPI master mode must be set and the flash memory must meet the following requirements:

- The flash memory must be able to operate at the clock frequencies given in Table 2.63.
- The flash memory must respond to the READ ID (0x 9F) command with a data value not equal to 0x 00 or 0x FF.
- The flash memory must understand the command FAST READ (0x 0B).

2.6.4 PLL primitives

A simplified PLL block diagram is shown in Figure 2.47. It contains the PLL core and some external circuitries. On the left side, the PLL input clock can be selected from the control register A to be either one of the clock signals mentioned in Section 2.6.1 on page 96, or it can be a clock signal from the user circuitry, namely USR_CLK_REF out of the FPGA fabric.

The PLL core clock outputs can be enabled via control register A bits and the locking state of the PLL as shown in Table 2.64.

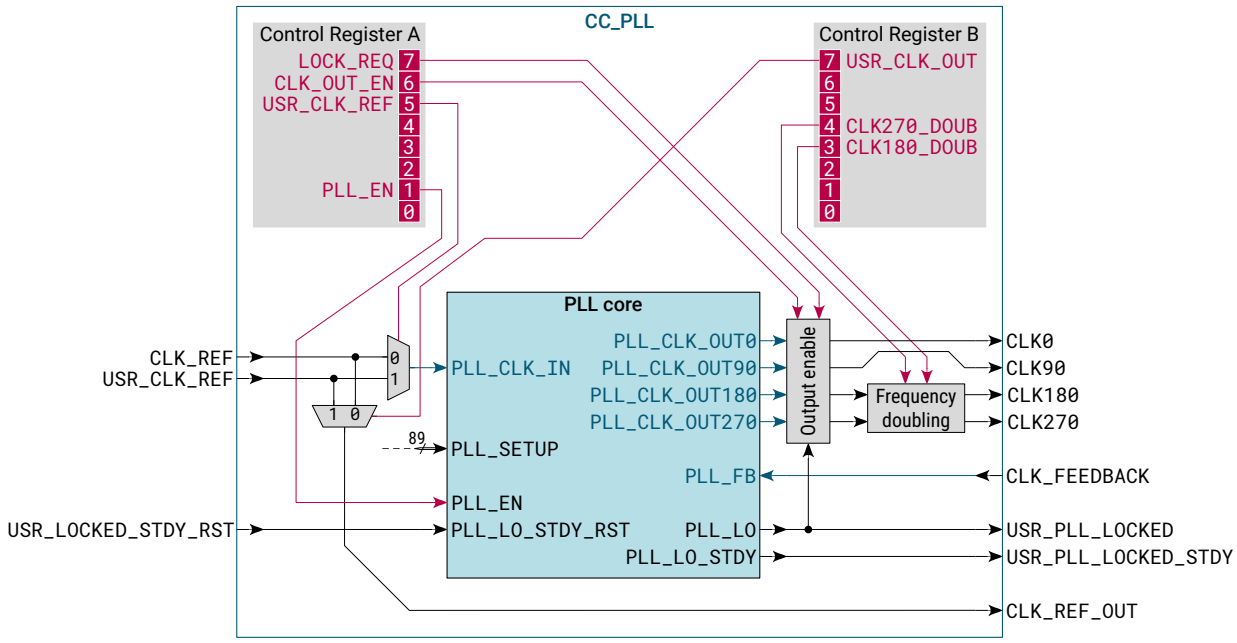


Figure 2.47: *Block diagram of the PLL primitive CC_PLL*

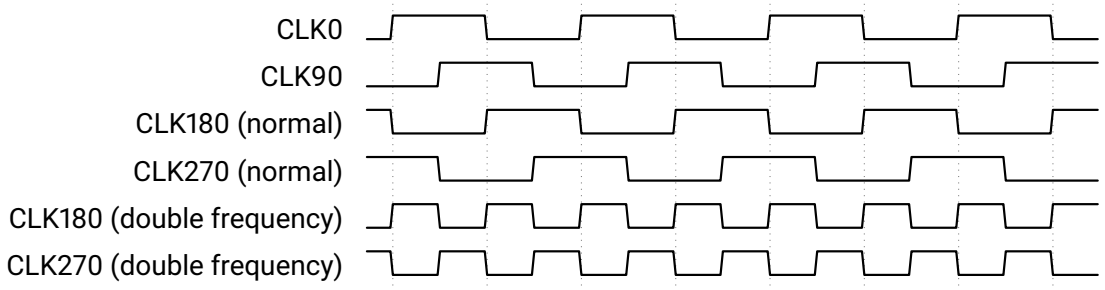


Figure 2.48: *CC_PLL output signals including frequency doubling on ports CLK180 and CLK270*

Table 2.64: *Activation of the PLL outputs*

CLK_OUT_EN	LOCK_REQ	PLL_LO	Clock output
0	X	X	✗ disabled
1	0	X	✓ enabled
1	1	0	✗ disabled
1	1	1	✓ enabled

The four clock phases are shown in Figure 2.48. The outpts CLK180 and CLK270 can be switched to double clock rate via control register B bits CLK180_DOU and CLK270_DOU. Figure 2.48 shows the clock edges of all CC_PLL clock outputs.

An advanced primitive of the PLL is shown in Figure 2.49. This is the same PLL, but the synthesis can optionally use primitive CC_PLL or CC_PLL_ADV for each individual PLL.

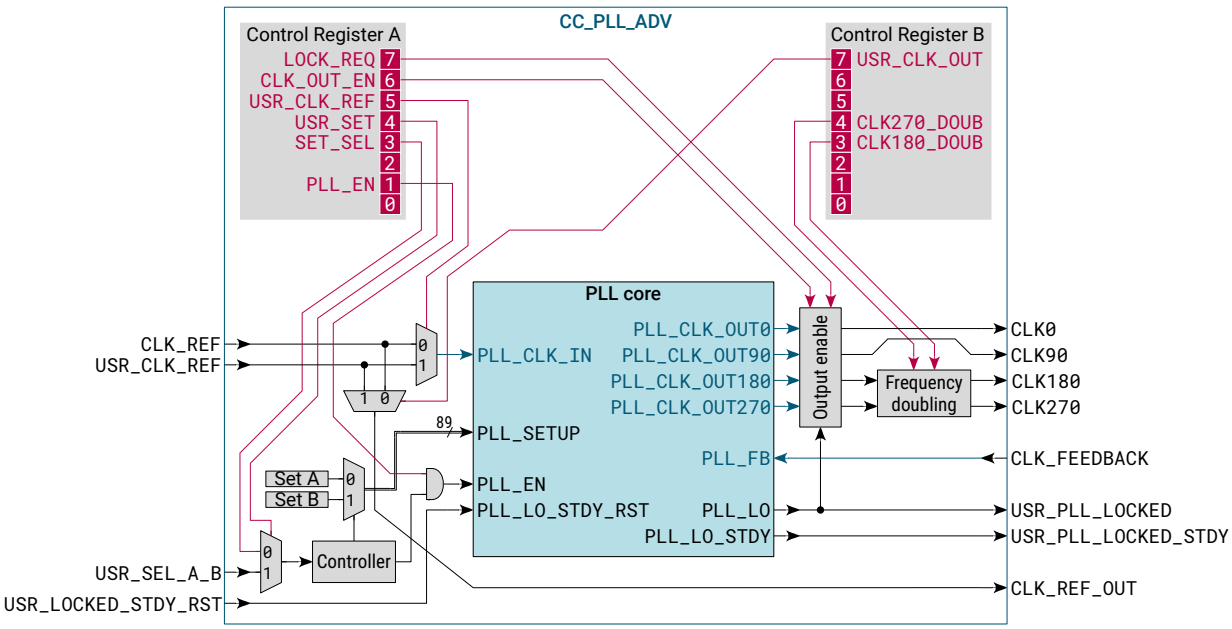


Figure 2.49: *Block diagram of the PLL primitive CC_PLL_ADV*

As a difference to CC_PLL, the advanced primitive CC_PLL_ADV contains the possibility to choose between two setups. When changing the PLL setup during operation, the controller automatically switches off the PLL during the switching phase and re-activates it then.

Table 2.65: *PLL control register A*

Bit	Name	Default	Function
1	PLL_EN	0	PLL enable: 0: PLL operation disabled 1: PLL operation enabled
3	SET_SEL	0	PLL setup selection: 0: PLL uses first setup 1: PLL uses second setup
4	USR_SET	0	User PLL setup enable: 0: PLL setup gets selected with bit SET_SEL in this register 1: PLL setup gets selected from user signal
5	USR_CLK_REF	0	User clock selection: 0: External reference clock (see clock sources on page 96) 1: Reference clock from user signal
6	CLK_OUT_EN	0	Clock output enable: 0: Clock outputs disabled 1: Clock outputs enabled
7	LOCK_REQ	0	Locked state required: 0: Clock outputs are enabled even when the PLL is not locked 1: Clock outputs require locked state

Table 2.66: *PLL control register B*

Bit	Name	Default	Function
3	CLK180_DOUB	0	Frequency doubling of PLL output CLOCK_180: 0: Normal signal CLOCK_180 1: Double frequency signal CLOCK_180
4	CLK270_DOUB	0	Frequency doubling of PLL output CLOCK_270: 0: Normal signal CLOCK_270 1: Double frequency signal and inverted CLOCK_270
7	USR_CLK_OUT	0	User clock selection for bypass output: 0: Bypass output feeds external clock from pin SER_CLK 1: Bypass output feeds user clock

2.7 Global Mesh Architecture

2.7.1 Overview of the Global Mesh Signal Injection

In each die there is a Global Mesh, each consisting of four signal traces that are used to distribute signals over the entire chip area. This is mainly used for clock signals, but is also useful for high fanout signals, e.g. enable or user reset.

Figure 2.50 shows that the signal feed into the Global Mesh is closely linked to the clock generation. The modules involved are described in detail in the following sections.

2.7.2 Clock Input Multiplexers CLKIN

The GateMate FPGAs CCGM1A1 / CCGM1A2 have four dedicated clock input pins CLK0..CLK3. These clock pins are the second pin functions of GPIO pins (see list on page 96). In addition, SER_CLK can also be used to feed a clock signal into the FPGA.

From these five input clocks the CLKIN module selects four output clocks as shown in Figure 2.51. Please note, that every FPGA die has its own CLKIN module but uses shared input clocks CLK0..CLK3 and SER_CLK. Clock inversion (180° phasing) is configurable to any CLKIN clock output CLK_REF[3:0]. These clocks are forwarded to the PLLs.

2.7.3 PLL

There are four PLLs, each with a PLL core as described in Section 2.6. Every clock ouput from the CLKIN multiplexers can be used as reference clock of a PLL. Figure 2.52 shows that these CLK_REF[3:0] clocks and alternative user clocks USR_CLK_REF[3:0] from CPE outputs RAM_O1 can be selected as PLL reference inputs.

Each PLL selects one of the input clocks and generates an output frequency that is output with four phases 0°, 90°, 180° and 270°. The logic block shown in Figure 2.52 contains setting options for clock output enable, locked state required and frequency doubling for 180°and 270°output clocks. In addition, both input clocks can be bypassed inside the PLL.

Table 2.67: *USR_CLK_REF⟨n⟩ sources*

PLL	CPE coordinate	CPE output
0	CPE(1,124)	RAM_O1
1	CPE(1,123)	RAM_O1
2	CPE(1,122)	RAM_O1
3	CPE(1,121)	RAM_O1

Please note that the coordinates in the Cartesian system are always given in the die orientation.

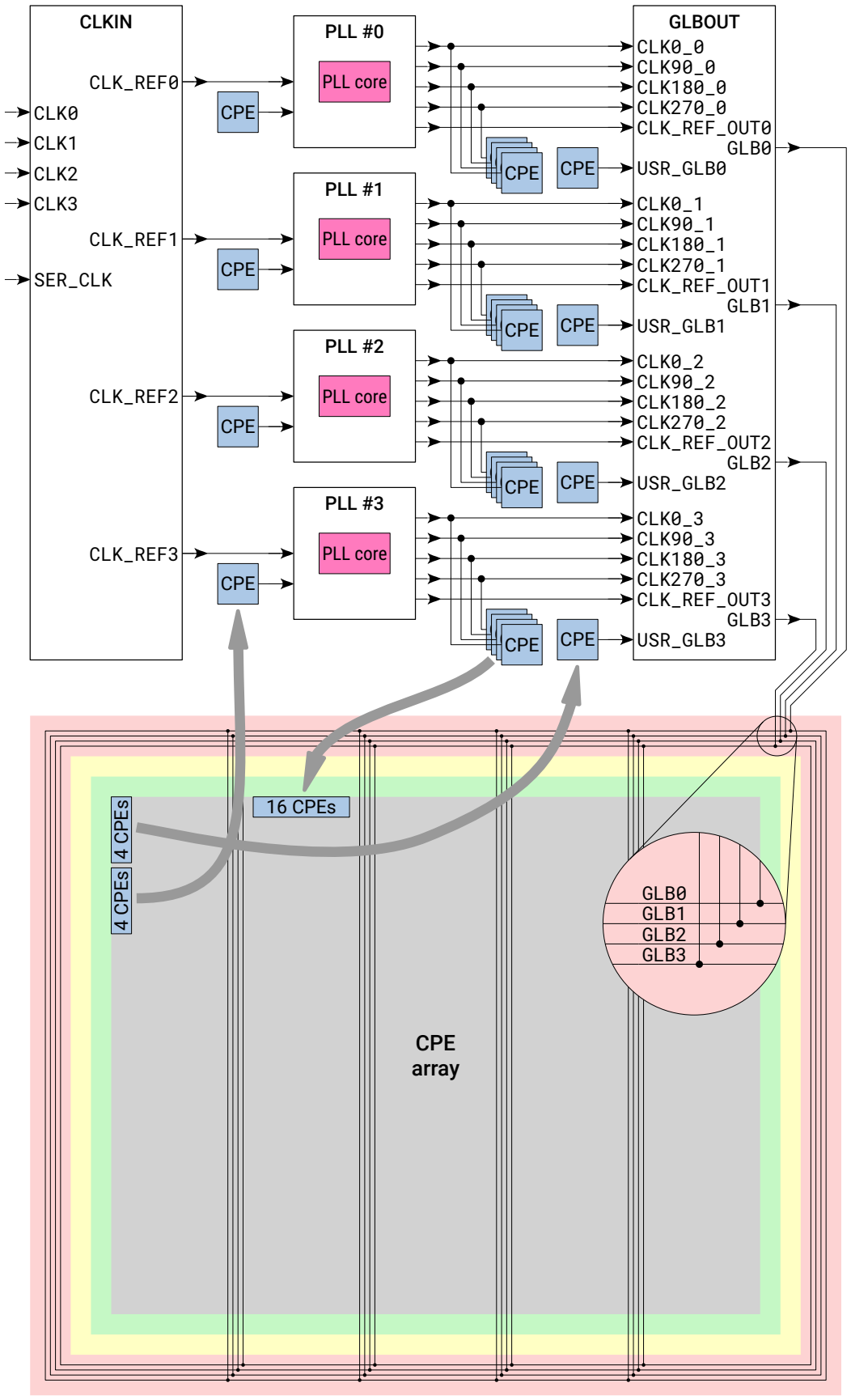


Figure 2.50: *Overview of the Global Mesh signal injection of a single die*

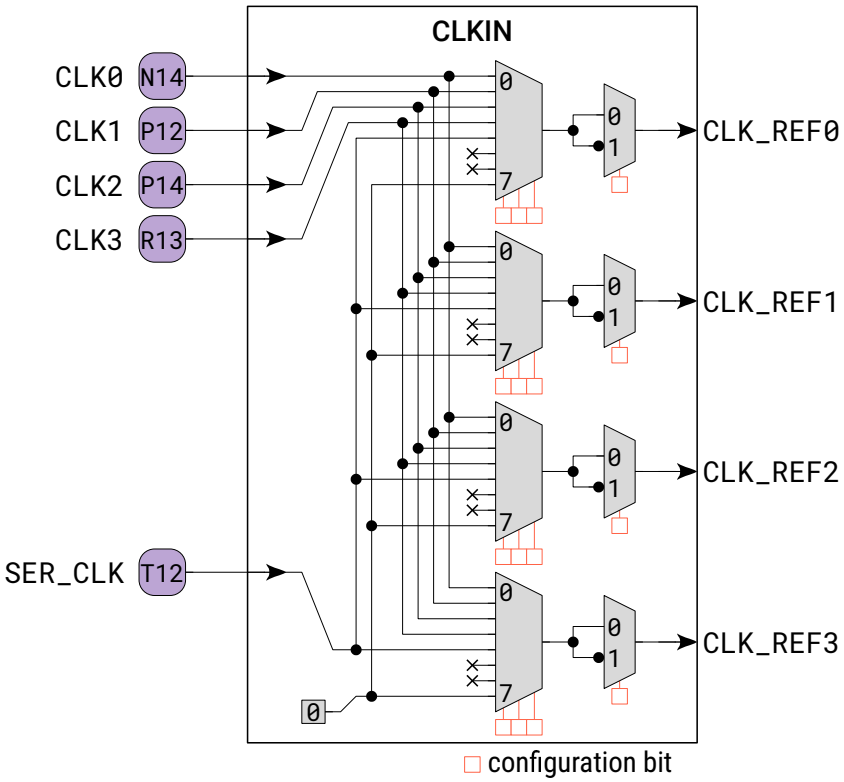


Figure 2.51: *Clock input multiplexers CLKIN*

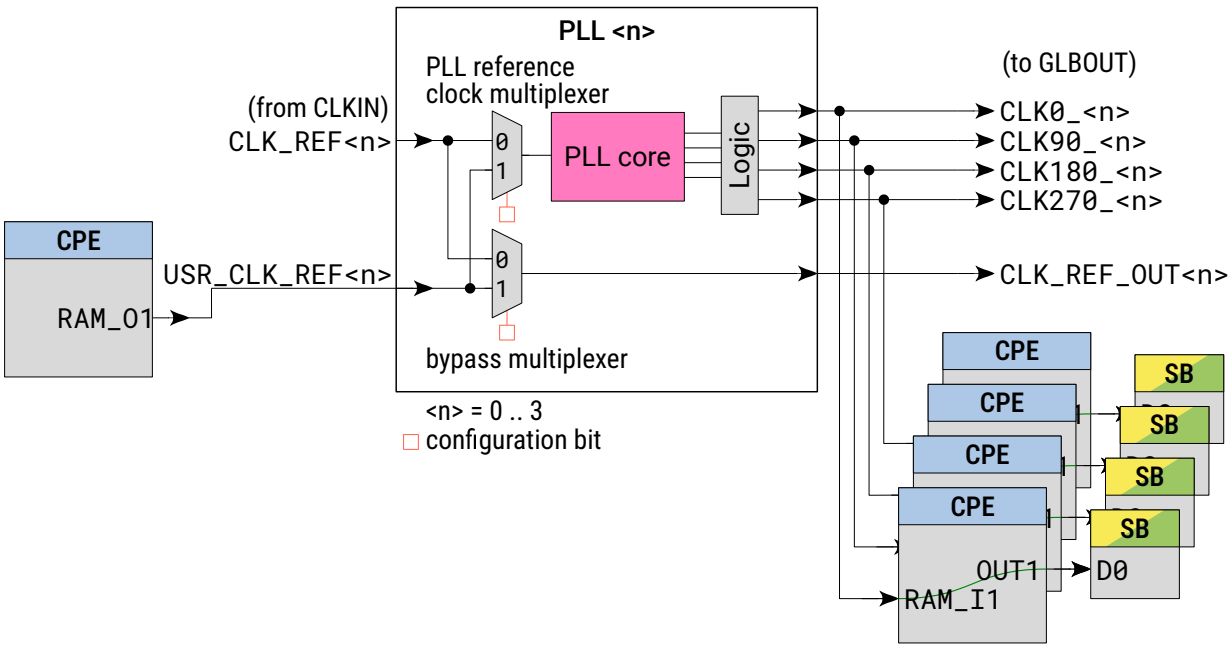


Figure 2.52: *PLL with input and output clocks*

The four PLL output clocks are connected to the GLBOUT module (see page 110) as well as to CPE inputs RAM_I1. From there, via CPE.OUT1, the clock signals are routed to Switch Boxes. Details to the CPE coordinates are given in Table 2.68.

Table 2.68: *CPE destinations of PLL clock output signals*

PLL	PLL clock output	CPE coordinate	CPE input / output
0	CLK0_0	CPE(39,128)	RAM_I1 / OUT1
0	CLK90_0	CPE(40,128)	RAM_I1 / OUT1
0	CLK180_0	CPE(41,128)	RAM_I1 / OUT1
0	CLK270_0	CPE(42,128)	RAM_I1 / OUT1
1	CLK0_1	CPE(43,128)	RAM_I1 / OUT1
1	CLK90_1	CPE(44,128)	RAM_I1 / OUT1
1	CLK180_1	CPE(45,128)	RAM_I1 / OUT1
1	CLK270_1	CPE(46,128)	RAM_I1 / OUT1
2	CLK0_2	CPE(47,128)	RAM_I1 / OUT1
2	CLK90_2	CPE(48,128)	RAM_I1 / OUT1
2	CLK180_2	CPE(49,128)	RAM_I1 / OUT1
2	CLK270_2	CPE(50,128)	RAM_I1 / OUT1
3	CLK0_3	CPE(51,128)	RAM_I1 / OUT1
3	CLK90_3	CPE(52,128)	RAM_I1 / OUT1
3	CLK180_3	CPE(53,128)	RAM_I1 / OUT1
3	CLK270_3	CPE(54,128)	RAM_I1 / OUT1

Please note that the coordinates in the Cartesian system are always given in the die orientation.

2.7.4 GLBOUT Multiplexers

The GLBOUT multiplexers are connected to five signals from a PLL (PLL output clock with four phases and bypassed clock) and four additional user signals USR_GLB[3:0] from dedicated CPE outputs RAM_O1 as shown in Table 2.69. Figure 2.53 shows, how these input signals can be selected to generate the four output signals GLB[3:0].

In most cases, GLB[3:0] are clock signals. But the user signals can also be other signals, typically high fanout signals, e.g. enable or user reset.

The GLB[3:0] are fed into the Global Mesh and can be accessed at many points with very low skew.

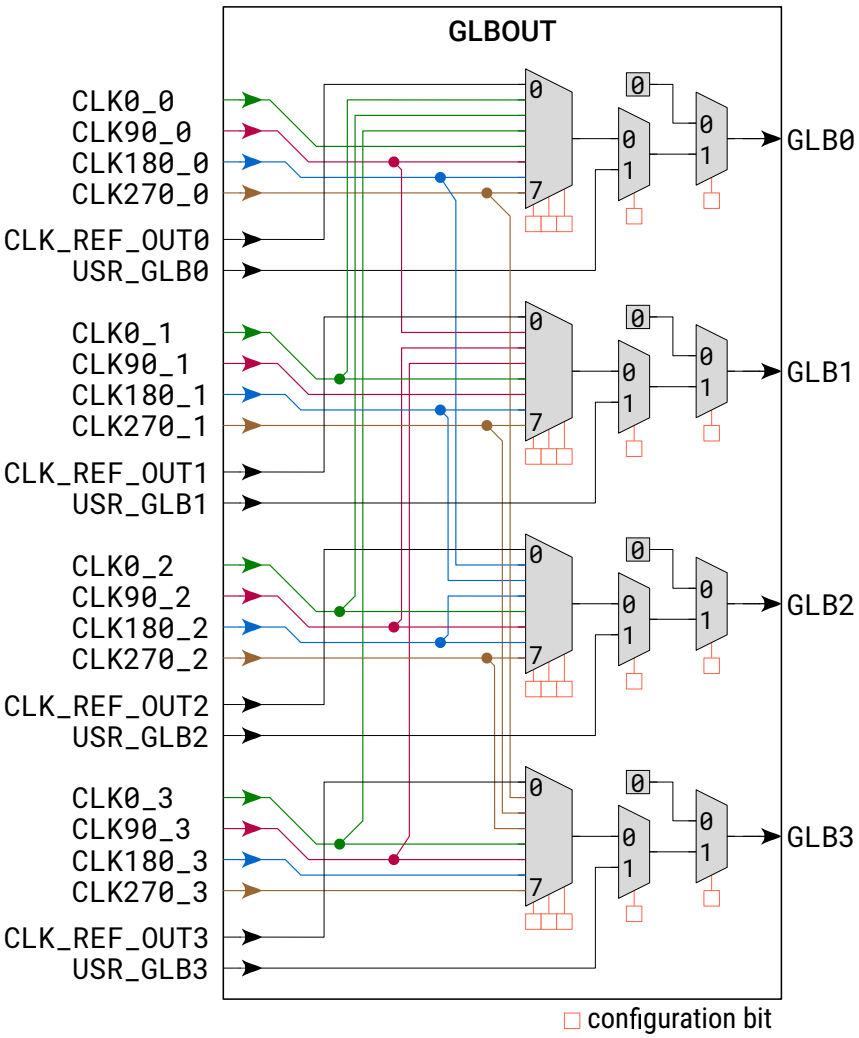


Figure 2.53: *GLBOUT multiplexers*

Table 2.69: *USR_GLB[3:0] sources*

Signal	CPE coordinate	CPE output
USR_GLB0	CPE(1,128)	RAM_O1
USR_GLB1	CPE(1,127)	RAM_O1
USR_GLB2	CPE(1,126)	RAM_O1
USR_GLB3	CPE(1,125)	RAM_O1

Please note that the coordinates in the Cartesian system are always given in the die orientation.

2.7.5 Use of the CC_BUFG Elements in HDL Design Sources

CC_BUFG elements are used to feed signals into the Global Mesh. As there are four nets in this mesh, a maximum of four CC_BUFG elements can be used in a design. If there are more, Place & Route will terminate with an error message. If there is no CC_BUFG element, the Global Mesh will not be used.

The synthesis distinguishes between clock signals and other signals to decide whether a CC_BUFG element is inserted or not. Therefore, some primitives have clock input ports, namely:

CC_IDDR.CLK	CC_BRAM_20K.A_CLK
CC_ODDR.CLK	CC_BRAM_20K.B_CLK
CC_DFF.CLK	CC_BRAM_40K.A_CLK
CC_DLT.G	CC_BRAM_40K.B_CLK
	CC_FIFO_40K.A_CLK
	CC_FIFO_40K.B_CLK

There are four steps from the user design to the FPGA bitstream:

1. Manual CC_BUFG instantiation:
   - The user can insert a CC_BUFG element to any signal. These signals will be routed over the Global Mesh. Up to four CC_BUFG elements can be inserted.
2. Synthesis option:
   - If the synthesis program gets started with the '-noclkbuf' option, no CC_BUFG elements will be inserted by the synthesis program.
   - If the option '-noclkbuf' is not used, the synthesis will insert automatically CC_BUFG elements to all nets which have dedicated clock inputs. Clock input nets are specified for primitive ports as listed above.
3. User intervention:
   - Before executing Place & Route, the user has to check how many CC_BUFG elements are included in the design.
   - If there are more than four CC_BUFG elements, the user has to repeat the first step and then, of course, the synthesis has to be started again with the '-noclkbuf' option. Otherwise, the Place & Route execution will fail with an error message.
4. Place & Route execution:
   - The assignment of GLB[3:0] signals to the CC_BUFG elements is done by the Place & Route software during the routing process. There is no need for the user to do this manually.

2.8 Clocking Schemes

2.8.1 The Methods of Clock Distribution

The Global Mesh can be used to distribute up to four signals over the entire chip area with small clock skew. The Global Mesh is typically used for clock signals. However, other signals, mostly high fan-out control signals like enable or user reset, can also be distributed with the Global Mesh instead. The edge frame of the chip contains a ring that distributes the Global Mesh signals (see also Figure 2.50 in previous Section 2.7.1). Four additional vertical trace structures bring the signals closer to the inside of the FPGA fabric.

This section describes several methods to distribute clock signals in the FPGA fabric. Essentially, the signals can be distributed to the user circuit in three ways:

1. Routing without using the Global Mesh. All signals are routed via the routing structure without any dedicated clock resources, as shown in Section 2.8.2.
2. Using the Global Mesh to distribute the Global Mesh signals as shown in Section 2.8.3.
3. Using CP-lines to distribute and forward clock and an associated enable signal over the entire chip area for clock-skew reduction, as shown in Section 2.8.4.

Figure 2.54 illustrates that there are four ways to pick up the Global Mesh signals:

Pick-up option 1: On the left edge every y position has a Left Edge Select (LES) element which can feed two signals

$$\begin{aligned} \texttt{LES}(-2,y).\texttt{CPE\_CINX} &\rightarrow \texttt{CPE}(1,y).\texttt{CINX} \\ \texttt{LES}(-2,y).\texttt{CPE\_PINX} &\rightarrow \texttt{CPE}(1,y).\texttt{PINX} \end{aligned}$$

with $1 \leqslant y \leqslant 128$ from the Global Mesh to CPE inputs.

Pick-up option 2: On the bottom edge every x position has a Bottom Edge Select (BES) element which can feed two signals

$$\begin{aligned} \texttt{BES}(x,-2).\texttt{CPE\_CINY2} &\rightarrow \texttt{CPE}(x,1).\texttt{CINY2} \\ \texttt{BES}(x,-2).\texttt{CPE\_PINY2} &\rightarrow \texttt{CPE}(x,1).\texttt{PINY2} \end{aligned}$$

with $1 \leqslant x \leqslant 160$ from the Global Mesh to CPE inputs.

Pick-up option 3: On all four edges every coordinate can feed all signals `GLB[3:0]` to a Big Switch Box (SB_BIG) stack as shown in Table 2.70. The specification $\texttt{SB\_BIG}(x,y,p)$ gives the x and y coordinates as well as the plane $1 \leqslant p \leqslant 12$. At the same coordinate (x,y) every Global Mesh signal is fed in parallel in three planes.

Pick-up option 4: Furthermore, the Global Mesh signals can be fed into the CPE array in four columns. Table 2.71 details which CPE coordinates and input signals are involved. Please notice that from the CPE input ports the signals are directly routed to Switch Box inputs:

$$\begin{aligned} \texttt{CPE.RAM\_I1} &\rightarrow \texttt{CPE.OUT1} \rightarrow \texttt{SB.D0} \\ \texttt{CPE.RAM\_I2} &\rightarrow \texttt{CPE.OUT2} \rightarrow \texttt{SB.D0} \end{aligned}$$

According to Table 2.72, the selection of the clocking scheme is made by the settings regarding `CC_BUFG` usage and the synthesis option '`-noclkbuf`'.

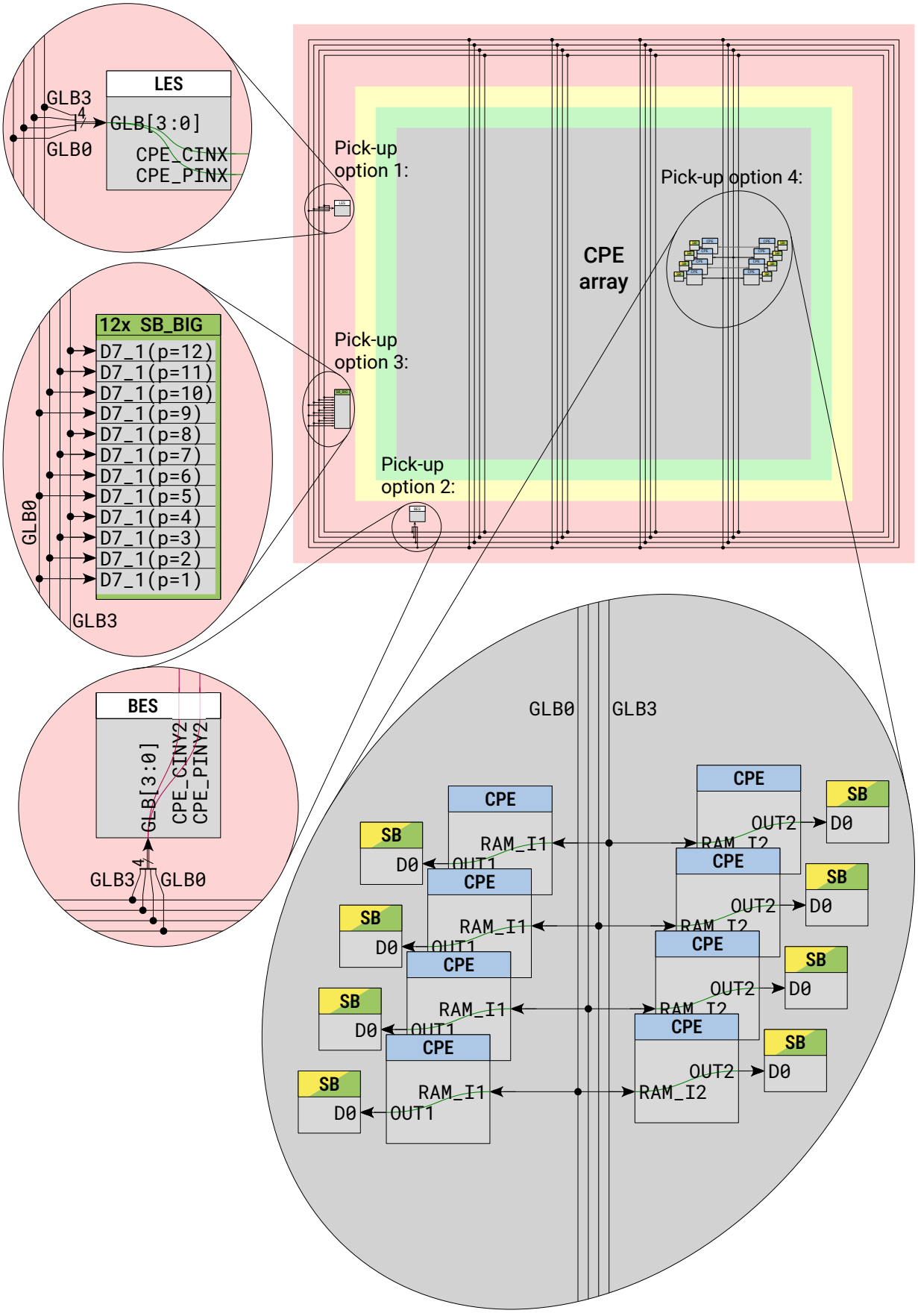


Figure 2.54: *Overview of signal extraction from the Global Mesh*

Table 2.70: *Connecting Global Mesh signals to Switch Boxes (pick-up option 3)*

FPGA edge	Signal	Target	Conditions
Left edge	GLB0	$\rightarrow$ SB_BIG$(x, y, \{1, 5, 9\})$.D7_1	$x = \begin{cases} 2 - (y \bmod 4) & ; 0 \leqslant y \leqslant 130 \\ -1 & ; y = -1 \end{cases}$
	GLB1	$\rightarrow$ SB_BIG$(x, y, \{2, 6, 10\})$.D7_1	
	GLB2	$\rightarrow$ SB_BIG$(x, y, \{3, 7, 11\})$.D7_1	
	GLB3	$\rightarrow$ SB_BIG$(x, y, \{4, 8, 12\})$.D7_1	
Bottom edge	GLB0	$\rightarrow$ SB_BIG$(x, y, \{1, 5, 9\})$.D7_2	$y = \begin{cases} 2 - (x \bmod 4) & ; 0 \leqslant x \leqslant 162 \\ -1 & ; x = -1 \end{cases}$
	GLB1	$\rightarrow$ SB_BIG$(x, y, \{2, 6, 10\})$.D7_2	
	GLB2	$\rightarrow$ SB_BIG$(x, y, \{3, 7, 11\})$.D7_2	
	GLB3	$\rightarrow$ SB_BIG$(x, y, \{4, 8, 12\})$.D7_2	
Right edge	GLB0	$\rightarrow$ SB_BIG$(x, y, \{1, 5, 9\})$.D7_3	$x = \begin{cases} 2 - (y \bmod 4) + 160 & ; 0 \leqslant y \leqslant 130 \\ 159 & ; y = -1 \end{cases}$
	GLB1	$\rightarrow$ SB_BIG$(x, y, \{2, 6, 10\})$.D7_3	
	GLB2	$\rightarrow$ SB_BIG$(x, y, \{3, 7, 11\})$.D7_3	
	GLB3	$\rightarrow$ SB_BIG$(x, y, \{4, 8, 12\})$.D7_3	
Top edge	GLB0	$\rightarrow$ SB_BIG$(x, y, \{1, 5, 9\})$.D7_4	$y = \begin{cases} 2 - (x \bmod 4) + 128 & ; 0 \leqslant x \leqslant 162 \\ 127 & ; x = -1 \end{cases}$
	GLB1	$\rightarrow$ SB_BIG$(x, y, \{2, 6, 10\})$.D7_4	
	GLB2	$\rightarrow$ SB_BIG$(x, y, \{3, 7, 11\})$.D7_4	
	GLB3	$\rightarrow$ SB_BIG$(x, y, \{4, 8, 12\})$.D7_4	

Table 2.71: *Connecting Global Mesh signals to CPEs (pick-up option 4)*

Signal		Target	
GLB0	$\rightarrow$	CPE$(x_0 - 3, y_0 + 10)$.RAM_I1	$x_0 = 33 + m \cdot 32 \;\; ; 0 \leqslant m \leqslant 3$
GLB1	$\rightarrow$	CPE$(x_0 - 3, y_0 + 11)$.RAM_I1	$= \{33, 65, 97, 129\}$
GLB2	$\rightarrow$	CPE$(x_0 - 3, y_0 + 12)$.RAM_I1	
GLB3	$\rightarrow$	CPE$(x_0 - 3, y_0 + 13)$.RAM_I1	
GLB0	$\rightarrow$	CPE$(x_0 + 2, y_0 + 10)$.RAM_I2	$y_0 = 1 + n \cdot 16 \;\; ; 0 \leqslant n \leqslant 7$
GLB1	$\rightarrow$	CPE$(x_0 + 2, y_0 + 11)$.RAM_I2	$= \{1, 17, 33, 49, 65, 81, 97, 113\}$
GLB2	$\rightarrow$	CPE$(x_0 + 2, y_0 + 12)$.RAM_I2	
GLB3	$\rightarrow$	CPE$(x_0 + 2, y_0 + 13)$.RAM_I2	

Table 2.72: *Control of the clocking scheme via CC_BUFG and CP-lines*

Number of clock signals with CC_BUFG	CP-lines	Section	Clocking scheme
0	disabled	2.8.2	Clock distribution via the routing structure
0	enabled	2.8.4.1	Intake CPE with chained CPE row
		2.8.4.2	Intake BES with chained BES row
		2.8.4.4	Intake LES with chained CPE row
1..4	disabled	2.8.3	Clock distribution via the Global Mesh
1..4	enabled	2.8.4.1	Intake CPE with chained CPE row
		2.8.4.3	Unchained BES row with Global Mesh pick-up
		2.8.4.2	Intake BES with chained BES row
		2.8.4.4	Intake LES with chained CPE row

2.8.2 Clock Distribution via the Routing Structure

The most general clocking scheme is the routing of clock signals through the routing structure. In this scheme, no clock signals may be in the Global Mesh and Place & Route must be started with the '`--clk_CP`' option. Any number of clock domains can be realized in any design.

However, the traces in the Global Mesh can distribute any other signals. They can be accessed with the pick-up options 3 (SB_BIG) and 4 (CPE) which are shown in Figure 2.54. The edge elements LES and BES are not available.

- With regard to electromagnetic interference (EMI), this clocking scheme is to be preferred. However, since no dedicated clocking resource is involved, the routing of clock signals via CPE and routing architecture might lead to high clock skew. This problem is addressed and cared about by the Place & Route software.
- On the other hand, this clocking scheme reduces the maximum clock frequency. If this is a problem, the following sections show possible solutions.

2.8.3 Clock Distribution via the Global Mesh

When using this scheme, the shortest path from any point in the Global Mesh to the user circuit is used to route the clock signal into the CPE array. Pick-up option 3 (SB_BIG) shown in Figure 2.54 is used.

Prerequisite for this clocking scheme is that clocks are in the Global Mesh and Place & Route is started with the '`--clk_CP`' option.

If not all traces in the Global Mesh are assigned with clock signals, free traces can distribute arbitrary other signals. Pick-up options 3 (SB_BIG) and 4 (CPE) are used to access these signals.

A maximum of four signals are allowed in the Global Mesh. If there are more, Place & Route will terminate with an error message.

- Since the Global Mesh signals are distributed simultaneously over the entire chip area, this clocking scheme leads to poorer EMI characteristics.
- On the other hand, Global Mesh signals have fast interconnection to arbitrary locations in the FPGA die, reducing the clock skew.

2.8.4 Clock Distribution via CP-lines

CP-lines can be used for fast clock routing in the entire chip area. This can be combined with clock distribution over the Global Mesh or the clock signals can be provided via the routing structure.

There are several ways to feed clock signals to the user circuit via CP-lines. In the following sections it is assumed that the user circuit occupies a rectangular area in the CPE array. This area is highlighted in the following Figures 2.55 to 2.58.

A clock signal and its associated enable signal are picked up by a so-called *intake element* and distributed via CP-lines. It is irrelevant whether one or both signals are in the Global Mesh or reach the intake element via the routing structure.

In the FPGA only one clock / enable signal pair can be routed via CP-lines. The enable signal is optional.

Prerequisite for this clocking scheme is that Place & Route is started with the '`++clk_CP`' option.

The principle of clock distribution via CP-lines can be differentiated into four types:

1. Intake CPE with chained CPE row (Section 2.8.4.1 on page 118)
2. Intake BES with chained BES row (Section 2.8.4.2 on page 119)
3. Unchained BES row with Global Mesh pick-up (Section 2.8.4.3 on page 120)
4. Intake LES with chained CPE row (Section 2.8.4.4 on page 121)

2.8.4.1 Intake CPE with chained CPE row

This clocking scheme inserts a horizontal CPE chain from $(x_0 - 1, y_0 - 1)$ to $(x_1, y_0 - 1)$ below the user circuit in its rectangular area with the lower left corner (x_0, y_0) and the upper right corner (x_1, y_1) (striped area in Figure 2.55) and forwards the clock signals via CP-lines in y direction. This allows a minimal clock skew in x and y direction. The intake CPE $(x_0 - 1, y_0 - 1)$ can be fed from the Global Mesh or from any signal in the routing structure.

Prerequisite for this clocking scheme is that at least one horizontal CPE row under the user circuit is not occupied.

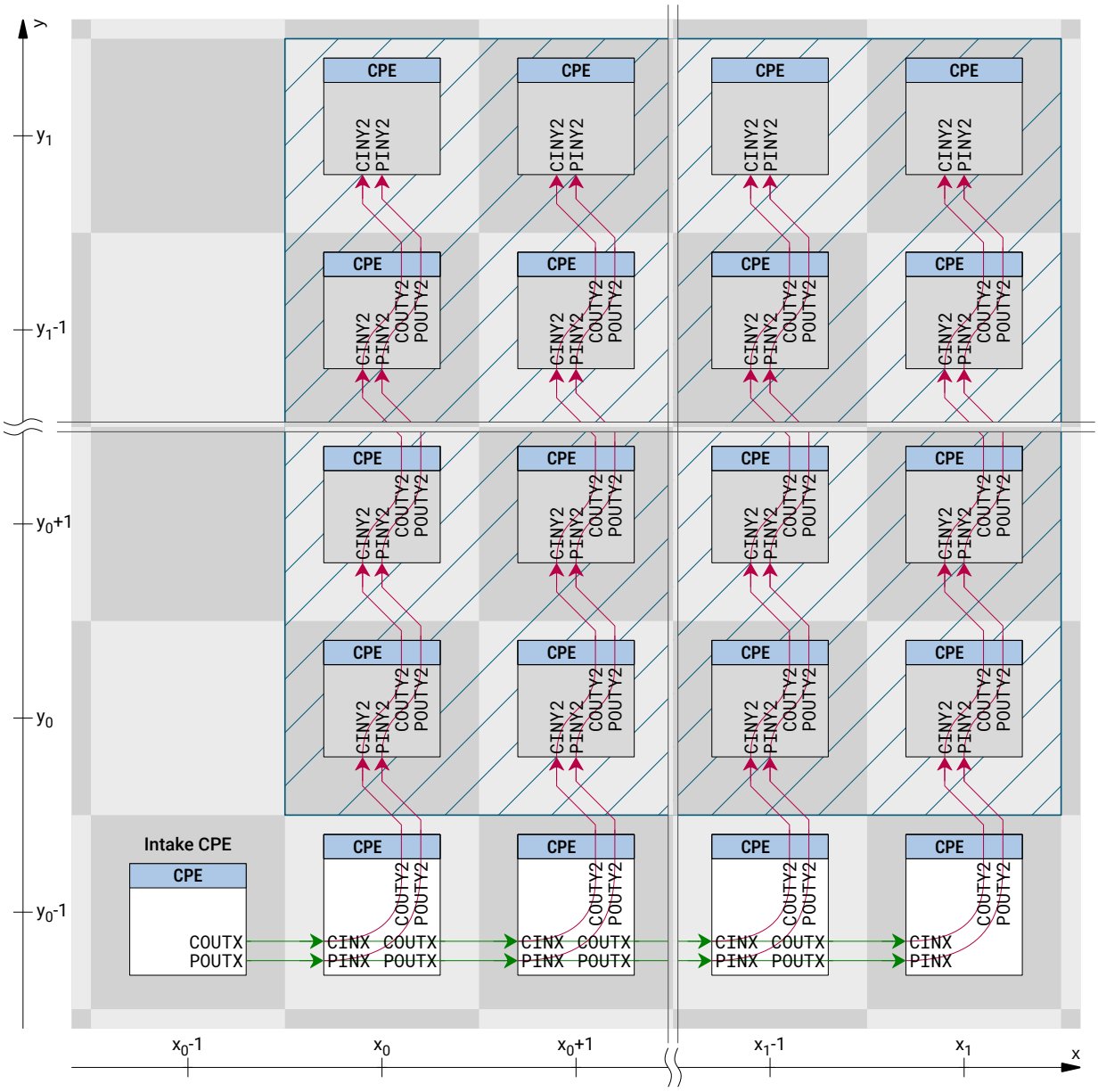


Figure 2.55: *Intake CPE with chained CPE row (die view)*

2.8.4.2 Intake BES with chained BES row

If there is no space below the user circuit (see striped area with the lower left corner $(x_0, 1)$ and the upper right corner (x_1, y_1) in Figure 2.56) to insert the additional CPE chain, clock and optional enable signal can be fed into the intake Bottom Edge Select (BES) element $(x_0, -2)$. From there, the signals are continued to coordinate $(x_1, -2)$.

The intake BES $(x_0, -2)$ can route Global Mesh signals as well as any signal from the routing structure to the CPE array.

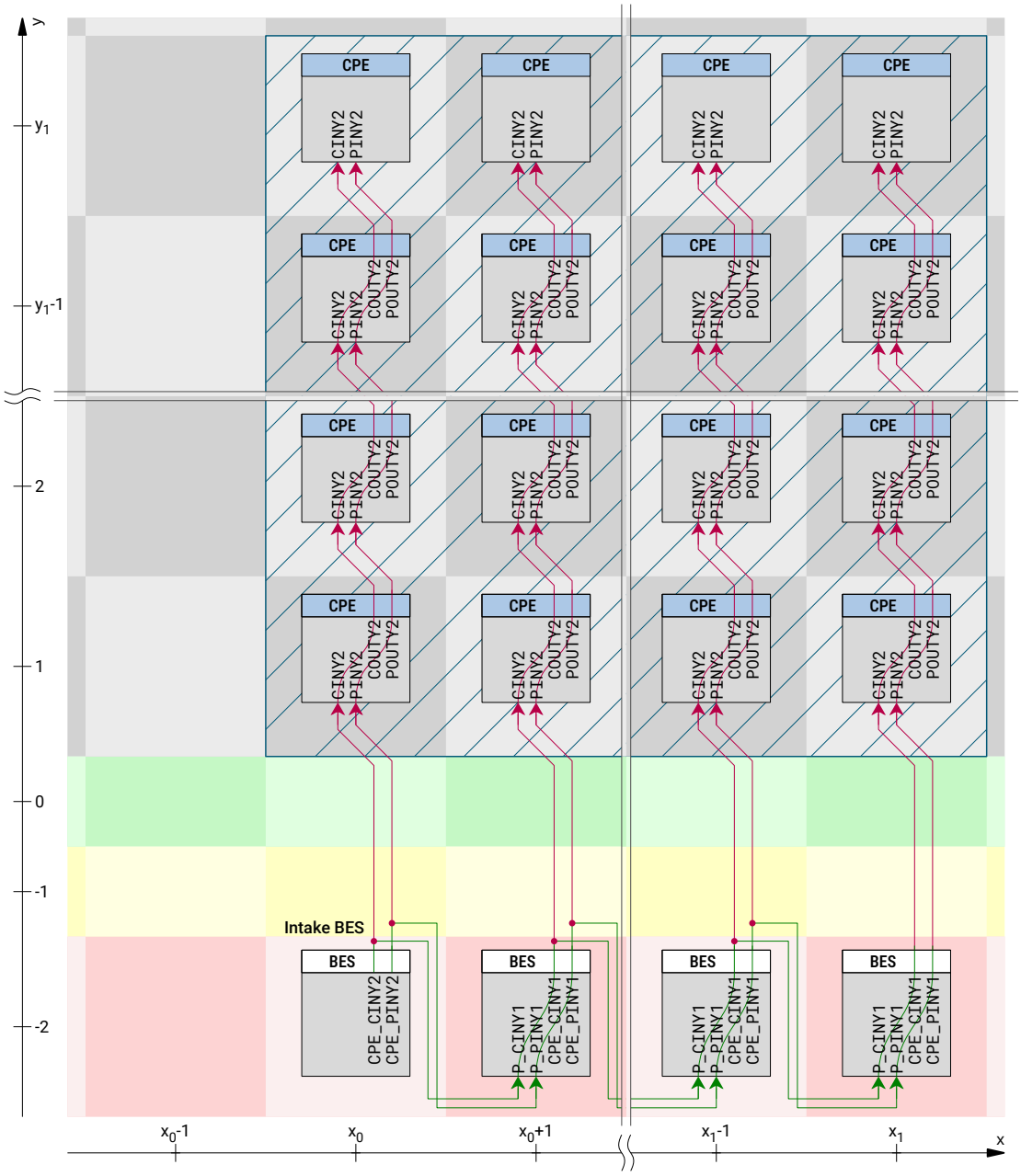


Figure 2.56: *Intake BES with chained BES row (die view)*

2.8.4.3 Unchained BES row with Global Mesh pick-up

As a variant of the previous Section 2.8.4.2, each one of the BES elements from $(x_0, -2)$ to $(x_1, -2)$ can pick up two Global Mesh signals as shown in Figure 2.57. In this case, clock skew occurs only in y direction.

Deviating from Figure 2.57, it is also possible to pick up only one of the two signals directly from the Global Mesh and chain the other signal via the BES elements as shown in Figure 2.56.

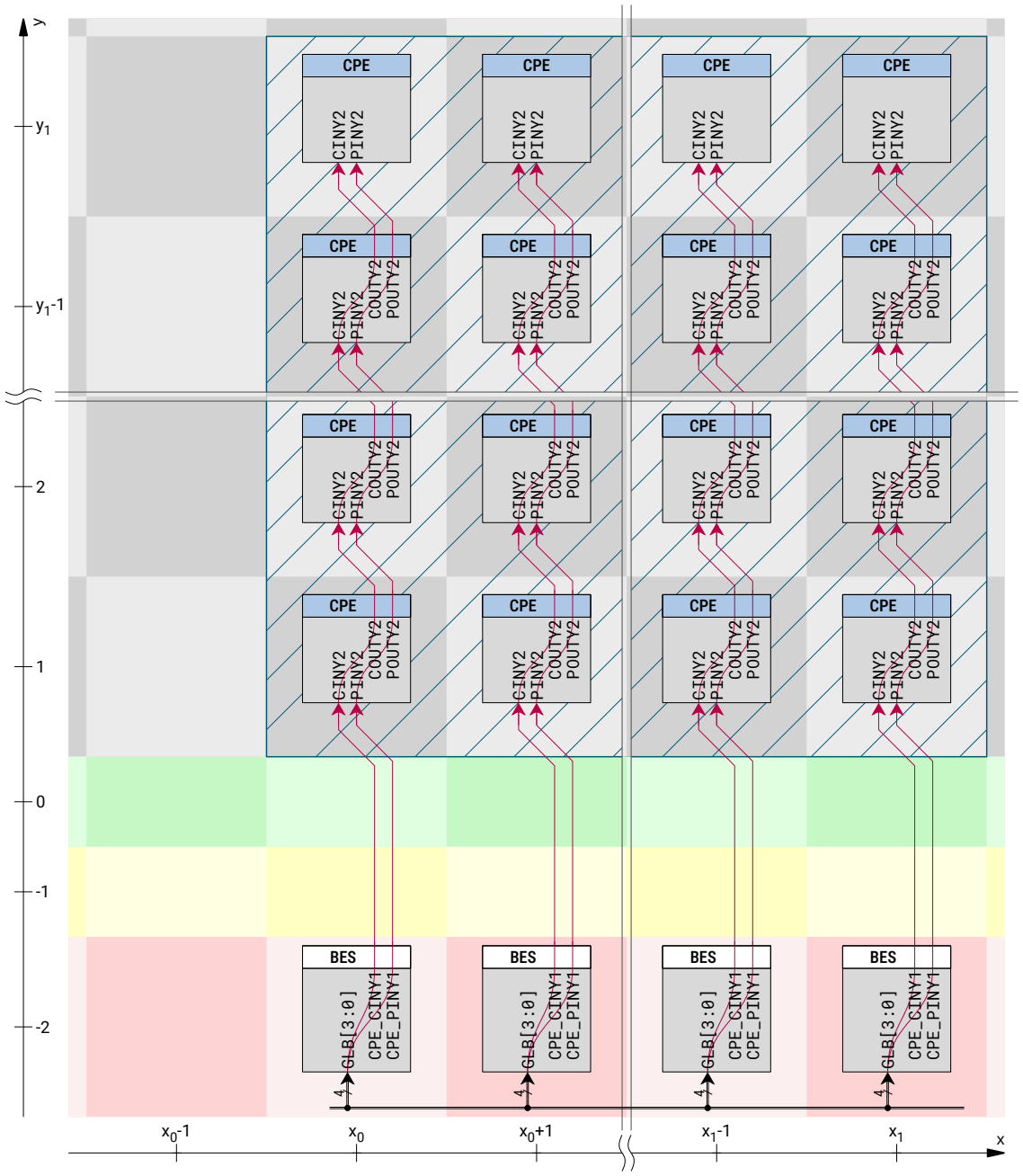


Figure 2.57: *Unchained BES row with Global Mesh pick-up (die view)*

2.8.4.4 Intake LES with chained CPE row

If the user circuit in its rectangular area with the lower left corner $(1, y_0)$ and the upper right corner (x_1, y_1) abuts the left edge of the CPE array (striped area in Figure 2.58), i.e. due to its size or placement in the CPE array, the intake Left Edge Select (LES) element $(-2, y_0 - 1)$ forwards the signal in x direction through the additional horizontal CPE chain from $(1, y_0 - 1)$ to $(x_1, y_0 - 1)$ below the user circuit. The intake LES can be fed from the Global Mesh or from any signal in the routing structure.

Prerequisite for this clocking scheme is that at least one horizontal CPE row under the user circuit is not occupied.

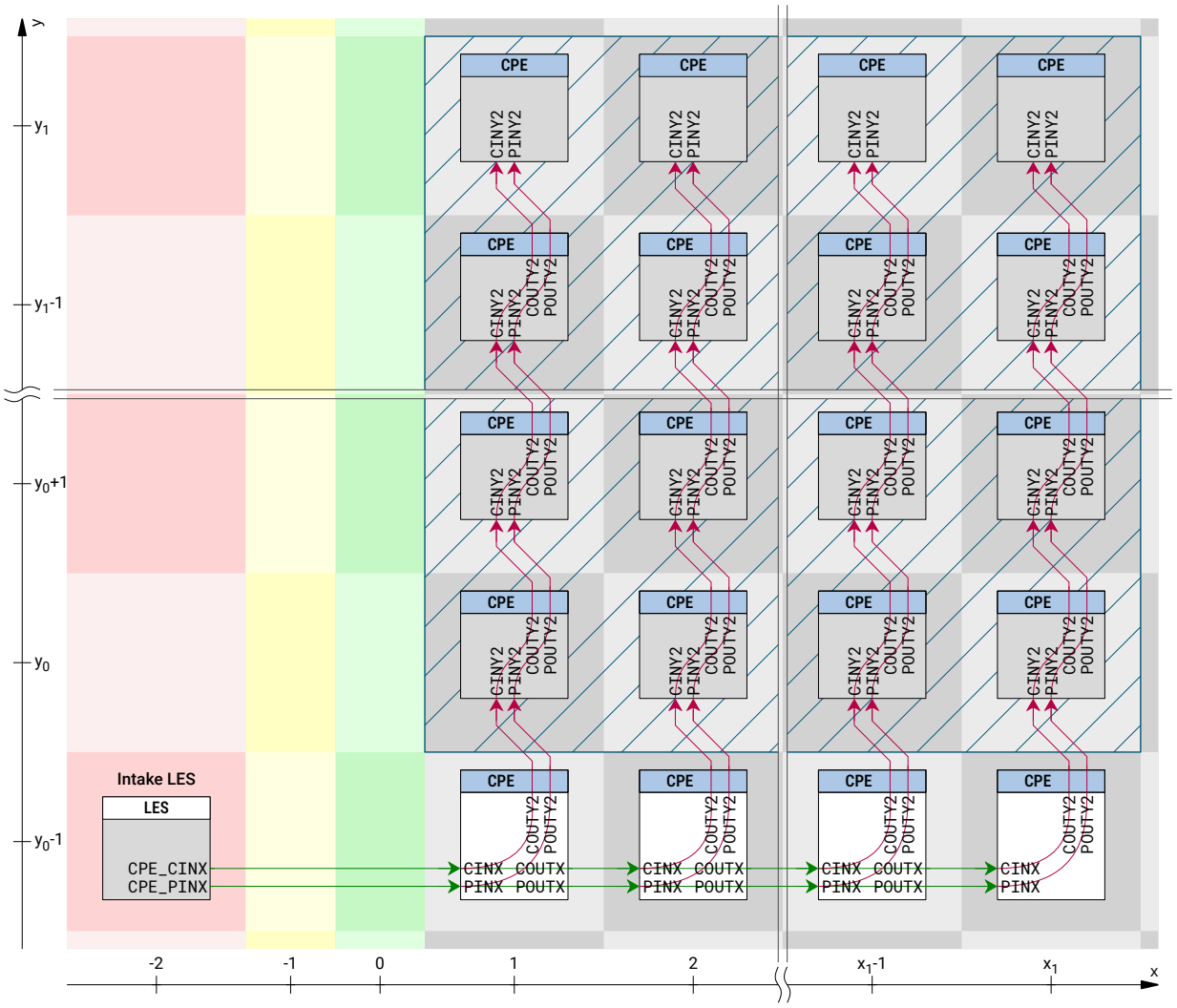


Figure 2.58: *Intake LES with chained CPE row (die view)*

2.9 Routing Structure

2.9.1 Switch Boxes

All functional elements of the GateMate™ FPGA are interconnected by the routing structure. This consist of so-called *Switch Boxes (SBs)* mainly and is supplemented by additional *Input Multiplexers (IMs)* and *Output Multiplexers (OMs)*. All Switch Boxes are arranged in a matrix $(x, y) = (-1, -1) \ldots (162, 130)$ as shown in Figure 2.59.

Every second coordinate is populated by Switch Boxes. There are two types of Switch Boxes which differ only in the span they can forward into the routing matrix. Mainly, Switch Boxes connect bidirectional to other Switch Boxes in horizontal and vertical direction. Big Switch Boxes connect six other Switch Boxes in every direction and Small Switch Boxes only two others. Direction change from vertical to horizontal and vice versa

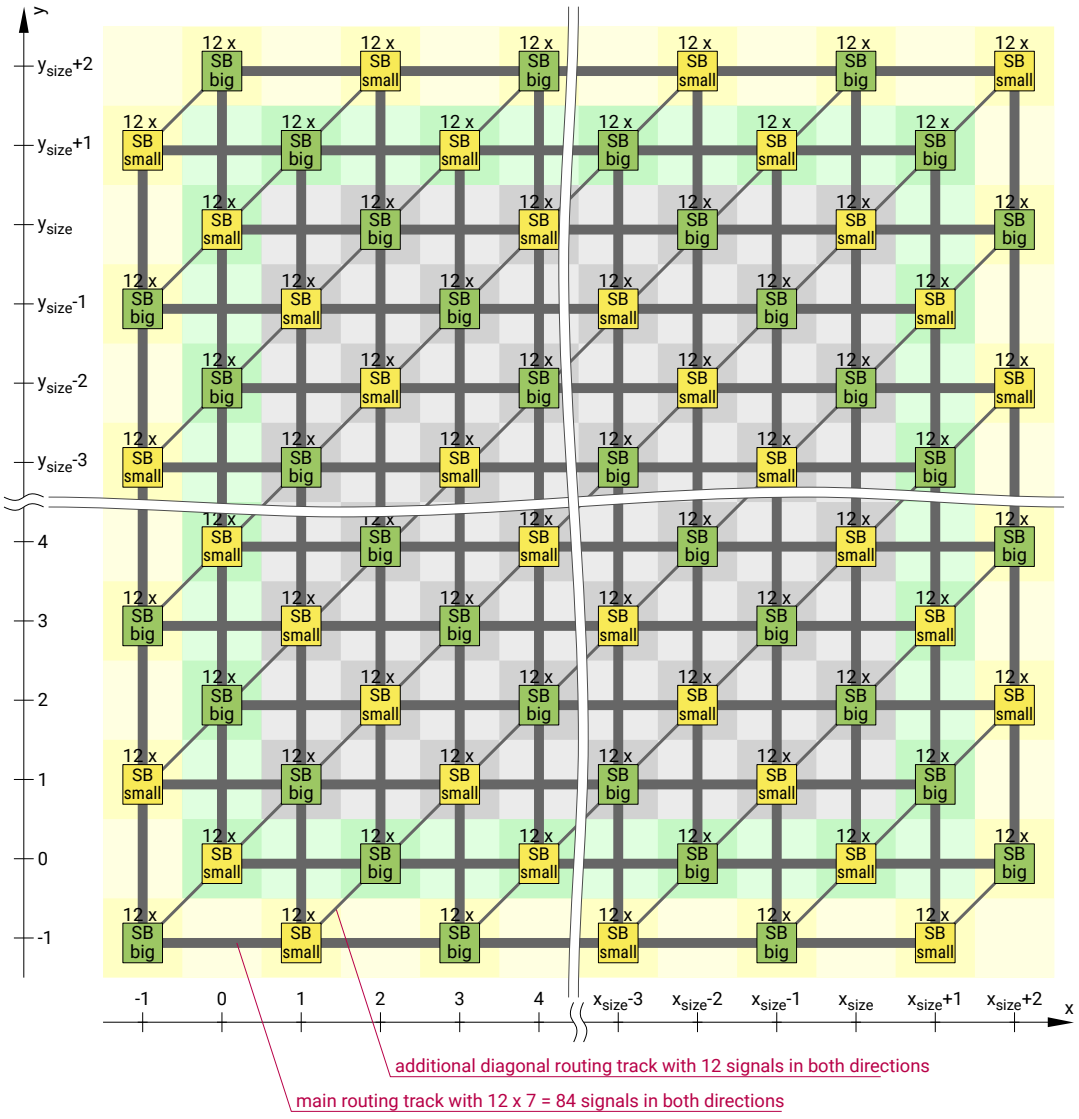


Figure 2.59: *Switch Box (SB) interconnection scheme (die view)*

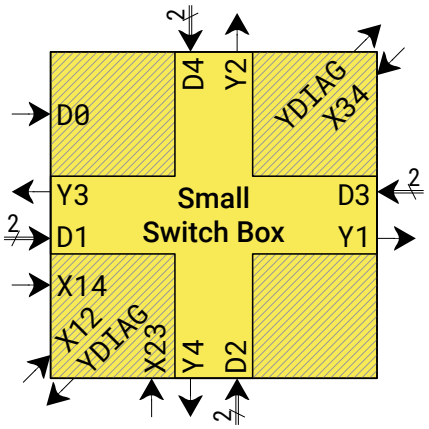


Figure 2.60: *Small Switch Box*

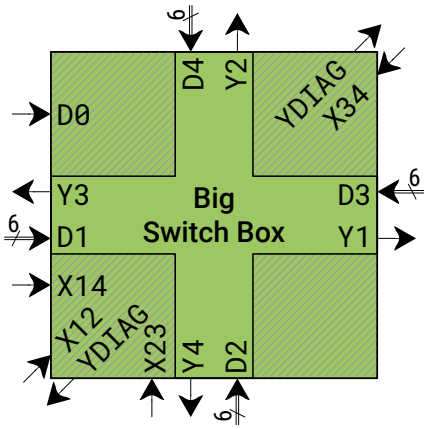


Figure 2.61: *Big Switch Box*

is possible. From an arbitrary point (x_n, y_m) all coordinates in row y_m or column x_n are reachable.

Furthermore, diagonal neighbors to upper right (x_n+1, y_m+1) and lower left (x_n-1, y_m-1) coordinate can be connected. This allows change to neighbored rows or columns.

Signals are feed into the routing structure either directly from CPE outputs or, for more flexibility, through Output Multiplexers (OMs). GPIO input signals are connected to the

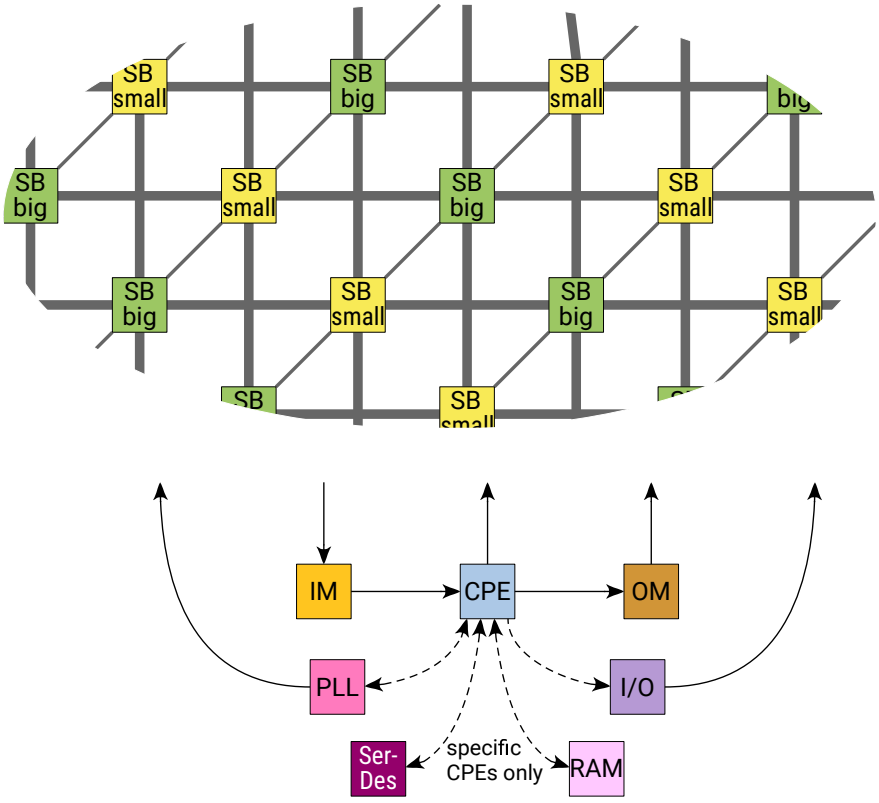


Figure 2.62: *CPE connections to the routing structure*

Switch Boxes as well.

The Switch Boxes decouple all signals through Input Multiplexers (IMs) to the CPEs.

2.9.2 Die-to-Die Connections

The die-to-die connections of the CCGM1A2 are electrically implemented directly on the semiconductor. They are not made via the interposer. Thus the routing structures of the two dies are directly interconnected as shown in Figure 2.63. There is a total of 1088 die-to-die connections in each signal direction.

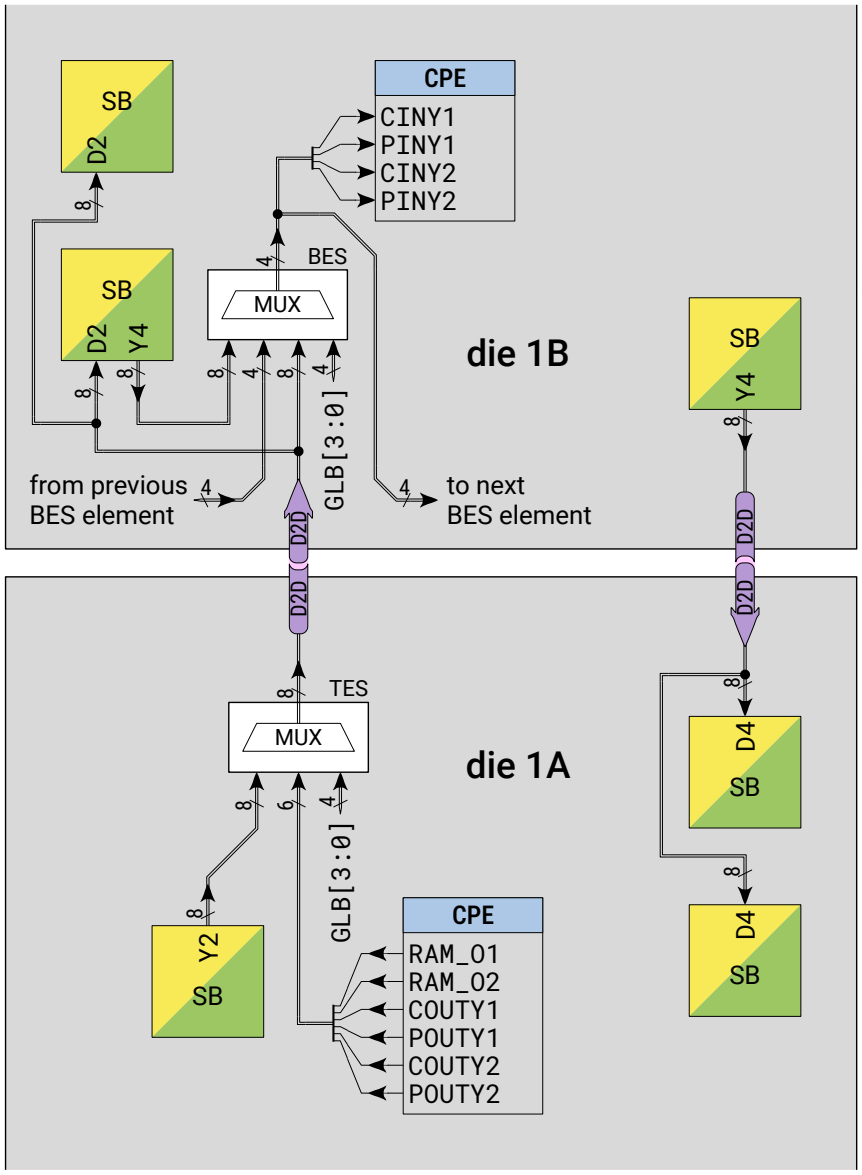


Figure 2.63: *Simplified routing concept of the die-to-die connections of CCGM1A2*

2.10 Power Supply

The GateMate™ FPGAs CCGM1A1 / CCGM1A2 have a single power supply for the chip core and GPIO power supply pins for each GPIO bank. There are some more power pins for dedicated parts of the FPGA:

VDD: Core supply voltage.

VDD_CLK: Supply voltage for the input pins SER_CLK, SER_CLK_N, RST_N and POR_ADJ [3].

VDD_PLL: Supply pin for PLLs.

VDD_SER: Supply pin for the SerDes module.

VDD_SER_PLL: Supply pin for the SerDes PLL.

VDD_EA, VDD_EB, VDD_NA, VDD_NB, VDD_SA, VDD_SB, VDD_WA, VDD_WB and VDD_WC: Supply pins for GPIO banks.

Core voltage can be chosen to select the performance mode:

Low power mode:	$\mathrm{VDD} = 0.9\,\mathrm{V} \pm 50\,\mathrm{mV}$
Economy mode:	$\mathrm{VDD} = 1.0\,\mathrm{V} \pm 50\,\mathrm{mV}$
Speed mode:	$\mathrm{VDD} = 1.1\,\mathrm{V} \pm 50\,\mathrm{mV}$

VDD_PLL has the same voltage range as VDD. VDD_SER and VDD_SER_PLL have a voltage range from 1.0 V ± 50 mV to 1.1 V ± 50 mV. All these voltages should be connected to noise filters.

The voltage range of GPIO banks and VDD_CLK can be set from 1.1 V to 2.7 V.

[3] POR_ADJ: CCGM1A1 only

Chapter 3

Workflow and Hardware Setup

3.1 FPGA Workflow

3.1.1 Overview

The workflow from design entry to the GateMate™ FPGA configuration file is shown in Figure 3.1.

Design entry is accomplished using entry tools or any Hardware Description Language (HDL). The Yosys Open SYnthesis Suite (Yosys) is used to perform Register Transfer Level (RTL) synthesis. It has extensive Verilog support. VHDL sources can be synthesized via GHDL through the ghdl-yosys-plugin. Other HDLs and tools with Verilog back-end can also be used. A guide to the synthesis options of Yosys for the GateMate techlib directory is described in Section 3.1.2.

Synthesis generates a gate-level representation of the design entry in form of a Verilog netlist of architecture-specific primitives. The resulting netlist can be used for post-synthesis simulation with any simulator. The required simulation models are part of the GateMate techlib directory in Yosys.

Furthermore, the netlist is passed to the Place & Route tool for architecture-specific implementation and bitstream generation. A netlist converter generates a generic netlist from the Yosys or legacy netlist. The first steps of Place & Route comprise procedures for speed or area optimization before mapping. After placement and routing, the static timing analysis (STA) might lead to further optimization steps and makes the Place & Route software an iterative process of constraint-driven re-placement and re-routing steps to finally achieve user requirements.

After successful placement and routing, a Verilog netlist with the associated Standard Delay Format (SDF) for timing-based post-implementation simulation are generated.

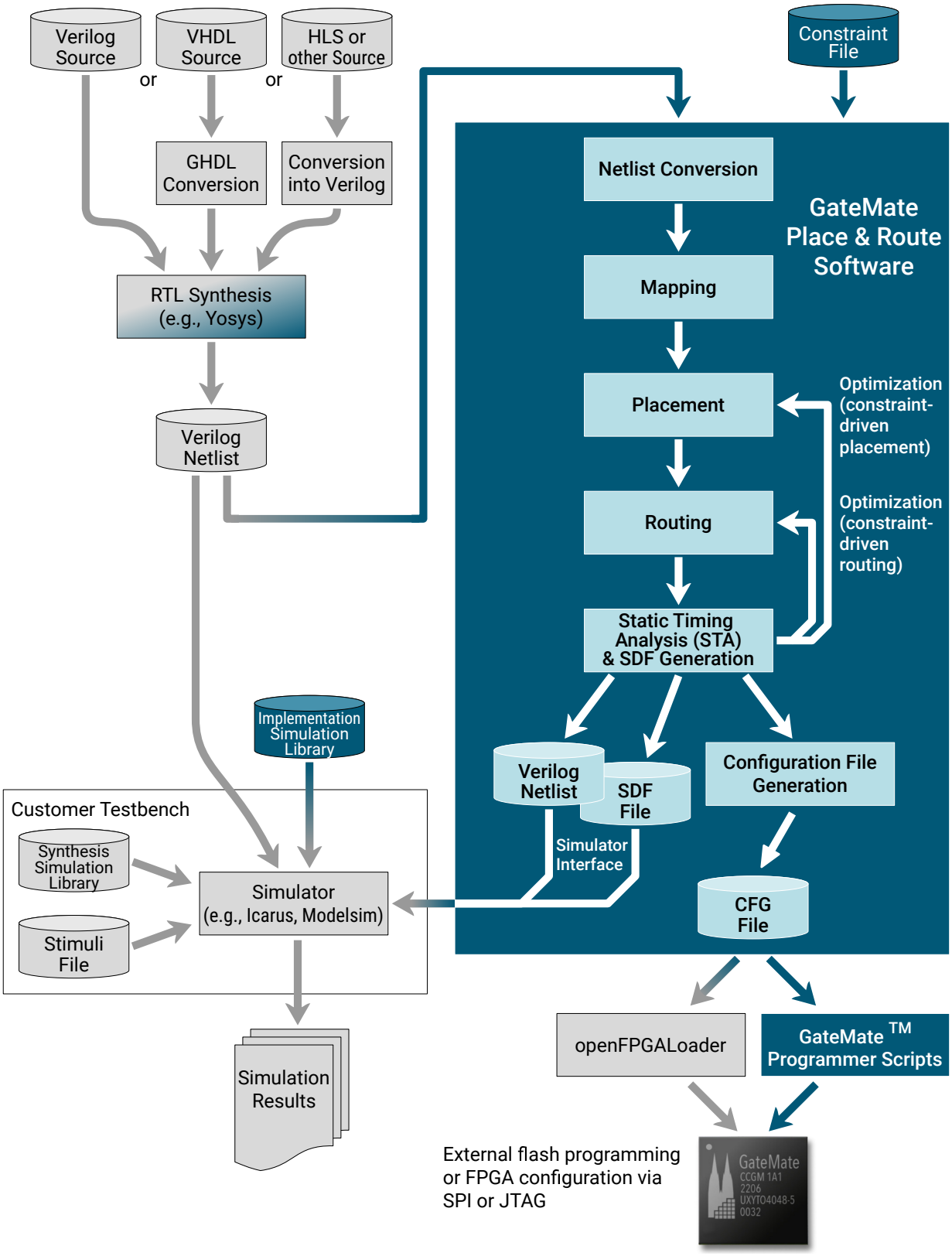


Figure 3.1: *GateMate™ FPGA workflow*

Simulation of the Verilog netlist requires a library of simulation models provided by Cologne Chip. Again, any simulator can be used.

The resulting configuration bitfile can either be written into a flash memory or directly to the FPGA configuration controller using it's JTAG or SPI interfaces. openFPGALoader has full support for GateMate™ FPGA configuration. Alternatively, Cologne Chip offers customizable scripts for configuration. A guide to programming external flash and FPGA configuration using openFPGALoader is described in Section 3.1.4.

3.1.2 Synthesis

This section describes the architecture-specific options of the `synth_gatemate` pass in Yosys to run synthesis.

`-top <module>`
: Uses the specified module as top-level module. In general, Yosys autodetects the top-level module within the set of input files.

`-noflatten`
: Omits flattening of the design before synthesis. Yosys automatically flattens the netlist for efficiency. This option gives per-module synthesis statistics that can be used to discover modules in need for optimization. Note that the Cologne Chip Place & Route accepts flattened netlists only.

The following commands disable architecture-specific cells. In order to be able to use the full functionality, none of these options should be set. These options generally slow down design performance and increase area usage.

`-nobram`
: Omits using `CC_BRAM_20K` or `CC_BRAM_40K` block RAM cells in the output netlist, forcing Yosys to build memories out of registers.

`-noaddf`
: Omits using `CC_ADDF` full adder cells in the output netlist.

`-nomult`
: Omits using `CC_MULT` multiplier cells in the output netlist.

`-nomx8, -nomx4`
: Omits using `CC_MX{8,4}` multiplexer cells in the output netlist.

`-noiopad`
: Disables I/O buffer insertion (useful for hierarchical or out-of-context flows).

`-noclkbuf`
: Disables automatic clock buffer `CC_BUFG` insertion.

Moreover, following synthesis-related options may be used to improve efficiency in terms of area and speed.

`-luttree` (experimental)
: This option enables LUT-tree mapping to `CC_L2T4` and `CC_L2T5` cells. Mapping to LUT-trees is omitted if this parameter is not specified. Yosys then defaults to traditional LUT mapping that results in less efficient implementation results during place and route.

`-retime`
: This option makes LUT mapping DFF-aware. It is able to remove or move registers to balance logic. Design performance benefits from that option.

In addition, there are options to call Yosys backends to write the netlist to a file.

`-vlog`
: Writes the design to the specified Verilog file. It is used as simulation netlist or input for the Cologne Chip Place & Route. Writing of an output file is omitted if this parameter is not specified.

`-json`
: Writes the design to the specified JSON file. It is used as input for Place & Route in `nextpnr` (future feature). Writing of an output file is omitted if this parameter is not specified.

3.1.3 Implementation

Place & Route Options

This section describes the available options of the Cologne Chip Place & Route to run implementation and bitstream generation. In general, the tool accepts Verilog netlists of architecture-specific primitives as generated in the `synth_gatemate` pass in Yosys. IO planning is handled with CCF constraint files.

Boolean options can be set with `+` and reset with `-` (`++` or `--` if using long option). Alternatively, `1` / `0`, `on` / `off` or `true` / `false` can be used for set and reset of an option as suffix. Option setting by suffix has priority over setting by `+` / `-`. E.g. `"-X on"` means option `X` is set.

`-h, --help`
: Displays help information and exits.

`-i <arg>, --input <arg>`
: Netlist input path and file name.

`-lib <arg>, --lib <arg>`
: Specifies the library format used in the input netlist. Available libraries are `{auto, ccag, xise, xvivado, yvivado}`.

`-o <arg>, --output <arg>`
: Output file name.

`-ccf <arg>, --ccf <arg>`
: CCF constraints input path and file name.

-A <arg>, --A <arg>
GateMate device number {1:CCGM1A1, 2:CCGM1A2} (default: 1).

-v, --verbose
Verbose mode.

-om <arg>, --fpga_mode <arg>
Operation mode {1:lowpower, 2:economy, 3:speed} (default: 3).

-tm <arg>, --time_mode <arg>
Timing mode {1:best, 2:typical, 3:worst} (default: 3).

The following commands control clock and optimization parameters.

-sf <arg>, --skew_factor <arg>
Skew violation factor of data path delay (default: 0.8).

+/-s, ++/--skew
Enable clock skew routing iteration (default: on).

+/-cCP, ++/--clk_CP
Use CP-lines for CPE inputs CLK and EN (default: on).

+/-gCP, ++/--gate_CP
Use CP-lines to build large gates using cascaded CPEs (default: on).

+/-cgb, ++/--clk_gl_bot
Force clocks over CP-lines from bottom edge (default: off).

+/-cgbc, ++/--gl_bot_chain
Force chaining instead of mesh in bottom edge (default: off).

+/-AF, ++/--AF
Combine adder and flip-flop functions in one CPE (default: off).

+/-dC, ++/--dual_CPE
Combine 2 gates into one CPE (default: on).

+/-sp, ++/--speed
Speed constraint (default: off).

+/-dl, ++/--del_latch
Delete latches with clamped G input (default: on).

+/-df, ++/--del_FF
Delete flip-flops with clamped set / reset inputs (default: on).

-pB <arg>, --place_box x<x1>y<y1>x<x2>y<y2>
Restricts the usable area in the CPE array, where {x1, x2} in [1..160] and {y1, y2} in [1..128].

The following commands control reports and output file parameters.

+/-ics, ++/--icarus_sdf
Generate Icarus Verilog compatible SDF (default: off).

+/-tp <arg>, ++/--time_pathes <arg>
Number of pathes in the timing report (default: 0).

+/-cdf, ++/--cdf
Write CDF file (default: on).

+/-pin, ++/--pin
Write PIN file (default: on).

+/-plc, ++/--plc
Write PLACE file (default: on).

+/-crf, ++/--crf
Generate cross reference file (default: off).

The following commands control global I/O-related parameters.

+/-CPE_in, ++/--CPE_inpin
Locate CPE next to FPGA input pin (default: off).

+/-CPE_out, ++/--CPE_outpin
Locate CPE next to FPGA output pin (default: off).

+/-uCIO, ++/--use_CFG_IOs
Use configuration I/Os as user GPIOs (default: off).

The following commands control configuration related parameters.

+/-pr, ++/--pwr_red
Reduce active power by setting unused Switch Box nodes inactive (default: on).

+/-m_spi <arg>, ++/--SPI_m_mode <arg>
SPI master settings {`00:disable, 11:single, 12:dual, 14:quad`} (default: 00).

+/-s_spi <arg>, ++/--SPI_s_mode <arg>
SPI slave settings {`0:disable, 1:single, 4:quad`} (default: 0).

CCF Contraint File Format

Entries in the CCF contraints file have the following format:

```
<pin-direction> "<pin-name>" Loc = "<pin-location>" | <opt.-constraints>;
```

Files are read line by line. Text after the hash symbol is ignored.

Available pin directions:

Pin_in defines an input pin.

Pin_out defines an output pin.

Pin_inout defines a bidirectional pin.

Additional constraints can be appended with the pipe symbol.

Available pin constraints:

SCHMITT_TRIGGER=*{true,false}*
Enables or disables Schmitt-trigger (hysteresis) option.

PULLUP=*{true,false}*
Enables or disables I/O pull-up resistor of nominal 50 kΩ.

PULLDOWN=*{true,false}*
Enables or disables I/O pull-down resistor of nominal 50 kΩ.

KEEPER=*{true,false}*
Enables or disables I/O keeper option.

SLEW=*{slow,fast}*
Sets slew rate to slow or fast.

DRIVE=*{3,6,9,12}*
Sets output driver strength to 3 mA . . 12 mA.

DELAY_IBF=*{0..15}*
Adds an additional delay of n times the nominal 50 ps to the input signal.

DELAY_OBF=*{0..15}*
Adds an additional delay of n times the nominal 50 ps to the output signal.

FF_IBF=*{true,false}*
Enables or disables placing of flip-flops in input buffer, if possible.

FF_OBF=*{true,false}*
Enables or disables placing of flip-flops in output buffer, if possible.

LVDS_BOOST=*{true,false}*
Enables increased LVDS output current of 6.4 mA (default: 3.2 mA).

LVDS_RTERM=*{true,false}*
Enables on-chip LVDS termination resistor of nominal 100 Ω, in input mode only.

Global I/O constraints can be set with the `default_GPIO` statement. It can be overwritten by individual settings for specific GPIOs.

```
default_GPIO | DRIVE=3; # sets all output strengths to 3mA, unless
   ↪ overwritten
```

3.1.4 Configuration

This section describes the options to configure the CCGM1A1 / CCGM1A2 and access external flash memory with openFPGALoader for the following boards and cables:

- **DS1003** – GateMate™ FPGA Evaluation Board
- **DS1002** – GateMate™ FPGA Programmer

Supported configuration files are bitfiles `*.bit` and it's ASCII equivalents `*.cfg` as generated from Cologne Chip Place & Route. The most useful options are listed as follows. Examples of how to use the tool are given below.

`-b, --board <arg>`
Specifies board and interface: '`-b gatemate_evb_jtag`' or '`-b gatemate_evb_spi`'.

`-c, --cable <arg>`
Specifies programmer cable: '`-b gatemate_pgm`'.

`--freq <arg>`
Use the specified frequency to shift the bitstream in Hz. Append 'k' to the argument for frequency in kilohertz, or 'M' for frequency in megahertz, e.g. '`--freq 10M`' (default: 6M).

`-f, --write-flash`
Writes the bitstream to an external flash memory.

JTAG Configuration

This mode performs an active hardware reset and writes the configuration into the FPGA latches via JTAG. The configuration mode pins `CFG_MD[3:0]` must be set to `0x C` (JTAG mode).

Program using the evaluation board:

```
$ openFPGALoader -b gatemate_evb_jtag <bitfile>.{cfg,bit}
```

Program using the programmer board:

```
$ openFPGALoader -c gatemate_pgm <bitfile>.cfg.bit
```

SPI Configuration

Performs an active hardware reset and writes the configuration into the FPGA latches via SPI. The configuration mode pins `CFG_MD[3:0]` must be set to `0x 4` (SPI passive mode).

Program using the evaluation board:

```
$ openFPGALoader -b gatemate_evb_spi -m <bitfile>.{cfg,bit}
```

Program using the programmer board:

```
$ openFPGALoader -b gatemate_pgm_spi -m <bitfile>.{cfg,bit}
```

JTAG Flash Access

It is possible to access external flashes via the internal JTAG-SPI bypass. The configuration mode pins `CFG_MD[3:0]` must be set to `0x C` (JTAG mode). Note that the FPGA will not start automatically.

Write to flash using the evaluation board:

```
$ openFPGALoader -b gatemate_evb_jtag <bitfile>.{cfg,bit}
```

Program using the programmer board:

```
$ openFPGALoader -c gatemate_pgm -f <bitfile>.{cfg,bit}
```

The 'offset' parameter can be used to store data at any point in the flash, e.g.:

```
$ openFPGALoader -b gatemate_evb_jtag -o <offset> <bitfile>.{cfg,bit}
```

SPI Flash Access

If the programming device and the FPGA share the same SPI signals, it is possible to hold the FPGA in reset and write data to the flash. The configuration mode can be set as desired. If the FPGA should start from the external memory after reset, the configuration mode pins `CFG_MD[3:0]` must be set to `0x 0` (SPI active mode).

Write to flash using the evaluation board:

```
$ openFPGALoader -b gatemate_evb_spi -f <bitfile>.{cfg,bit}
```

Program using the programmer board:

```
$ openFPGALoader -b gatemate_pgm_spi -f <bitfile>.{cfg,bit}
```

The 'offset' parameter can be used to store data at any point in the flash, e.g.:

```
$ openFPGALoader -b gatemate_evb_spi -o <offset> <bitfile>.{cfg,bit}
```

3.2 Hardware Setup

3.2.1 Power-on Reset

The GateMate™ FPGAs CCGM1A1 / CCGM1A2 have a reset pin RST_N with Schmitt-trigger characteristic and internal pull-up resistor. Signal polarity is active low.

The power-on reset (POR) module ensures a safe reset as soon as core voltage VDD and VDD_CLK supply voltages are stable. Figure 3.2 shows the POR block diagram. The POR module has the following features:

- Safe reset generation after stable VDD and VDD_CLK voltages.
- Voltage drop detection for VDD and VDD_CLK voltages.
- Adjustable VDD_CLK voltage threshold.
- No restrictions concerning order of voltage switch-on.
- Temperature compensated voltage reference.

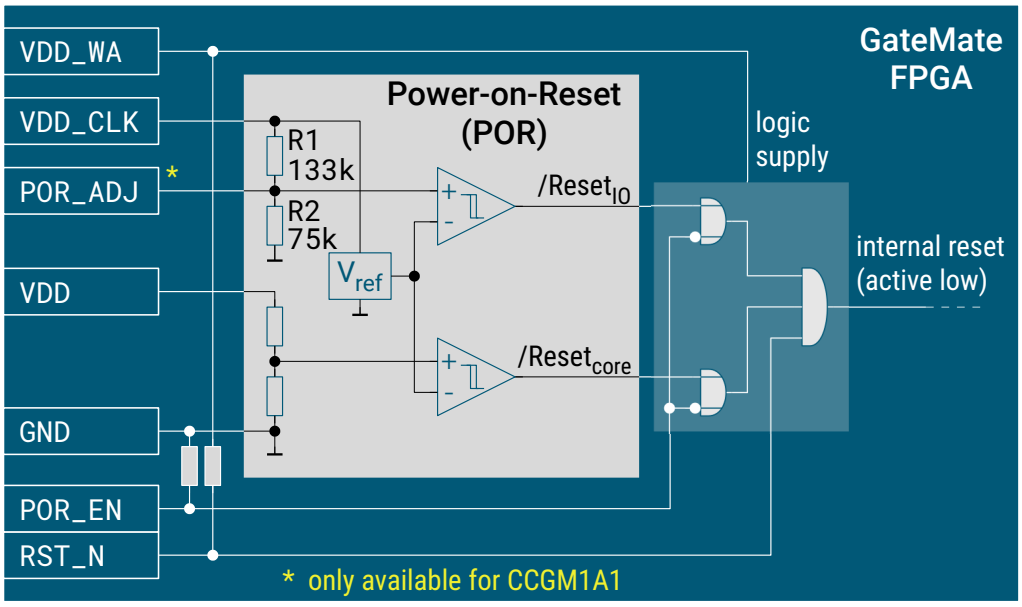


Figure 3.2: *Power-on reset (POR) module block diagram*

The POR must be enabled with POR_EN signal:

0: Reset state depends only on the RST_N signal.

1: FPGA changes from reset to operation state when RST_N is 1 and POR indicates VDD and VDD_CLK voltages are stable.

POR_EN has an internal pull-down resistor and can be left open if not used.

Figure 3.3 shows an exemplary supply voltage curve to explain the operation behavior of the POR reset output signals. Please see the electrical specification given in Table 4.7 on page 163 for exact threshold values in different operating cases.

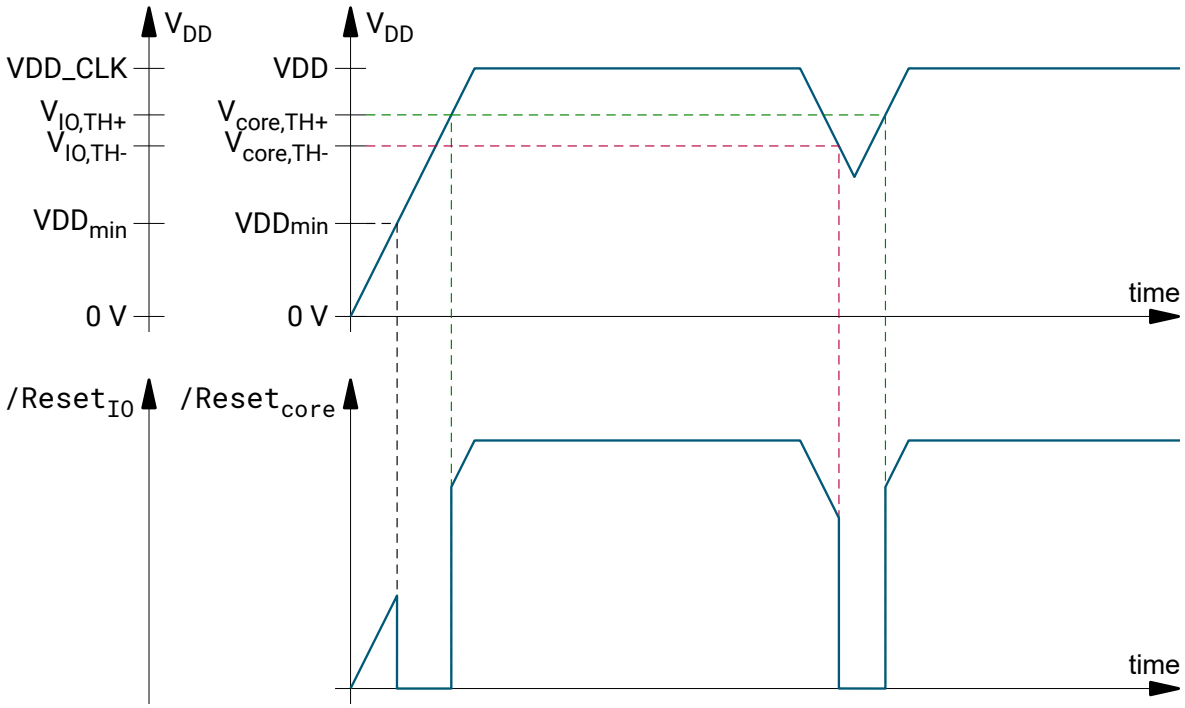


Figure 3.3: *Threshold principles of POR*

3.2.2 FPGA reset using the POR module

Using the POR module, in many cases no external components are required to ensure a safe reset of the GateMate FPGA. Figure 3.4 shows how to connect the reset pins in this case:

- VDD_WA and VDD_CLK must be connected to 1.1 . . 2.7 V supply (might be different).
- POR_EN has to be connected to VDD_WA.
- Pin RST_N must be connected to VDD_CLK.
- In normal operation, POR_ADJ should be left open. This pin is only available with CCGM1A1, but not with CCGM1A2. The optional voltage divider R3a and R3b is only used, if POR adjustment is required. This depends on the VDD_CLK level (see explanation below).

POR adjustment for CCGM1A1 (not available with CCGM1A2):

- With VDD_CLK in the range 1.8 . . 2.7 V (normal operation), POR_ADJ should be left open.
- R3b will increase the threshold level. Typically, this is not necessary within the specified VDD_CLK supply range. Nevertheless, reset state is permanently set with R3b $\leqslant 100\,\mathrm{k\Omega}$.
- R3a will decrease the threshold level. This is only necessary when VDD_CLK $< 1.8\,\mathrm{V}$ to avoid a permanent reset state. There will be no reset condition when R3a $\leqslant 100\,\mathrm{k\Omega}$.

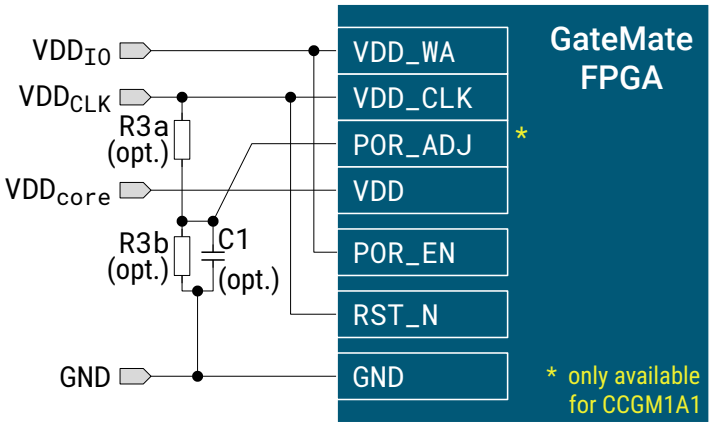


Figure 3.4: *FPGA reset using the POR module*

- For very low VDD_CLK voltages only a dynamic reset generation by means of R3a and C1 circuitry is possible. POR_ADJ is temporarily kept at ground level by C1. After charging C1 via R3a, the reset is released. The reset time can be approximated by

$$t_{reset} = \frac{\text{R3a} \cdot 75\,\text{k}\Omega}{\text{R3a} + 75\,\text{k}\Omega} \cdot \text{C1} \cdot \ln\left(\frac{\text{VDD\_CLK}}{\text{VDD\_CLK} - \frac{0.45\,\text{V}}{75\,\text{k}\Omega} \cdot (75\,\text{k}\Omega + \frac{\text{R3a} \cdot 133\,\text{k}\Omega}{\text{R3a} + 133\,\text{k}\Omega})}\right)$$

for R3a $< 10\,\text{k}\Omega$ this can be simplified to

$$t_{reset} \approx \text{R3a} \cdot \text{C1} \cdot \ln\left(\frac{\text{VDD\_CLK}}{\text{VDD\_CLK} - 0.45\,\text{V}}\right)$$

The reset time should be adjusted to the slope time of the VDD_CLK ramp-up.

3.2.3 FPGA reset without POR module

If the POR module is not to be used (see Figure 3.5), the following settings are required:

- VDD_WA and VDD_CLK must be connected to 1.1 . . 2.7 V supply (might be different).
- POR_EN has to be connected to GND or it can be left open due to the internal pull-down resistor.
- POR_ADJ can be left open.
- Pin RST_N must be connected to a logic or RC circuit to ensure a rising edge after stable power supply.

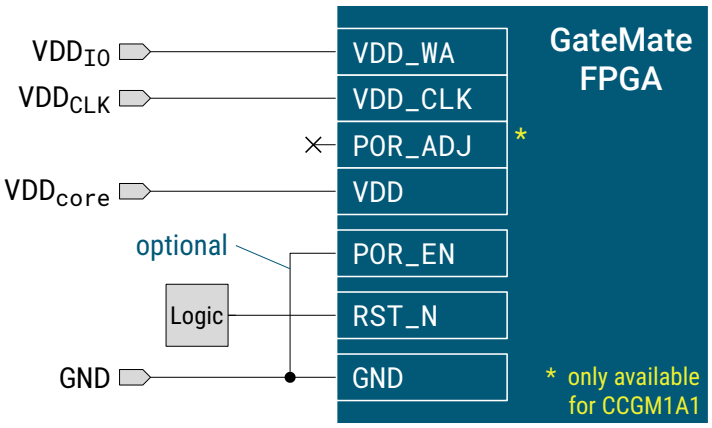


Figure 3.5: *FPGA reset driven by an external logic*

3.3 Configuration Procedure

3.3.1 Configuration Modes

The GateMate™ FPGA can be configured from four different sources:

1. The configuration process is controlled by the FPGA itself where it acts in SPI active mode, e.g., after reset. Configuration data is read from an external flash memory.
2. A processor unit operating as SPI in active mode feeds the configuration data to the FPGA which acts in SPI passive mode.
3. The JTAG interface of the FPGA connects to a JTAG test & programming adapter.
4. User configured logic of the FPGA can feed configuration data to the FPGA itself.

With the rising edge of the reset signal the configuration mode setup at pins `CFG_MD[3:0]` gets captured as shown in Table 3.1. These pins must always be connected to either `GND` or `VDD_WA`.

Table 3.1: *Configuration mode setup*

`CFG_MD[3:0]` *		Configuration mode	
`0x0`	`0b0000`	SPI Active Mode	CPOL = 0, CPHA = 0
`0x1`	`0b0001`	SPI Active Mode	CPOL = 0, CPHA = 1
`0x2`	`0b0010`	SPI Active Mode	CPOL = 1, CPHA = 0
`0x3`	`0b0011`	SPI Active Mode	CPOL = 1, CPHA = 1
`0x4`	`0b0100`	SPI Passive Mode	CPOL = 0, CPHA = 0
`0x5`	`0b0101`	SPI Passive Mode	CPOL = 0, CPHA = 1
`0x6`	`0b0110`	SPI Passive Mode	CPOL = 1, CPHA = 0
`0x7`	`0b0111`	SPI Passive Mode	CPOL = 1, CPHA = 1
`0xC`	`0b1100`	JTAG	

* Modes not mentioned in the list may not be selected and lead to a malfunction of the device.

3.3.2 Configuration Signals

Table 3.2: *Configuration signal bank*

Pin	Signal name	Reset category	I/O mode	I/O characteristic	Description
R5	CFG_MD0	4	I		Configuration mode bit 0
T5	CFG_MD1	4	I		Configuration mode bit 1
U4	CFG_MD2	4	I		Configuration mode bit 2
V4	CFG_MD3	4	I		Configuration mode bit 3
U1	POR_EN	2	I	pull-down	Enable power-on reset
V3	CFG_DONE	2	O	pull-down, low, driver strength 12 mA	Configuration done signal
V2	CFG_FAILED_N	2	I/O	pull-up, high, open drain, driver strength 12 mA	Configuration failed signal
N3	SPI_CS_N	3	I, I/O	pull-up, driver strength 12 mA	Configuration SPI chip select
N4	SPI_CLK	3	I, I/O	driver strength 12 mA	Configuration SPI clock
P2	SPI_D0	3	I, I/O	driver strength 12 mA	Configuration SPI data bit 0
P1	SPI_D1	4	I		Configuration SPI data bit 1
R2	SPI_D2	4	I		Configuration SPI data bit 2
R1	SPI_D3	4	I		Configuration SPI data bit 3
R4	SPI_FWD	2	O	low, driver strength 12 mA	Configuration SPI data forward
T3	JTAG_TMS	4	I		JTAG test mode select
R3	JTAG_TCK	4	I		Configuration JTAG clock
T2	JTAG_TDI	4	I		JTAG data input
U2	JTAG_TDO	2	O	low, driver strength 12 mA	JTAG data output

Reset categories:
0: No reset (supply or analog input pin)
1: Pin disabled, high-z
2: Pin characteristic already during reset state
3: Pin characteristic depends on the configuration mode (SPI master: I/O, else: input)
4: Input pin

3.3.3 SPI Configuration

The SPI bus of GateMate™ FPGA can be used to load the configuration either in SPI active or passive mode. Both modes use the same pins. Typically, a flash memory is connected to the SPI bus to provide the configuration data. Figure 3.6 shows a typical circuitry which uses only single-I/O or dual-I/O mode. The flash signals I/O_2 and I/O_3 can have further functions depending on the used device and should have pull-up resistors in most cases.

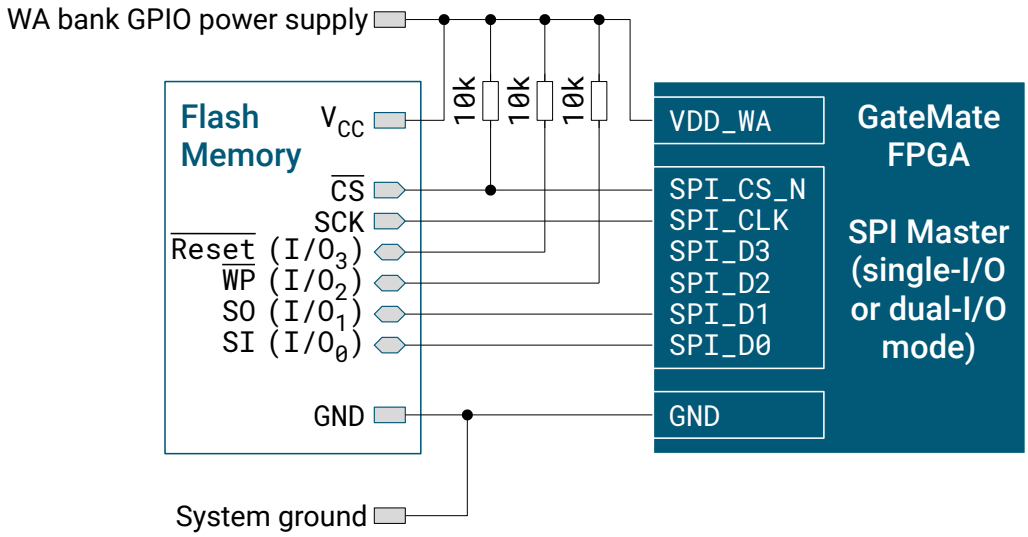


Figure 3.6: *Configuration data from flash memory in SPI single-I/O or dual-I/O mode*

When SPI quad-I/O mode is used, Figure 3.7 shows the typical circuitry. Now, all data lines are connected and no pull-up resistors are required. Depending on the used flash device, pull-up resistors are recommended at I/O_2 and I/O_3 signals.

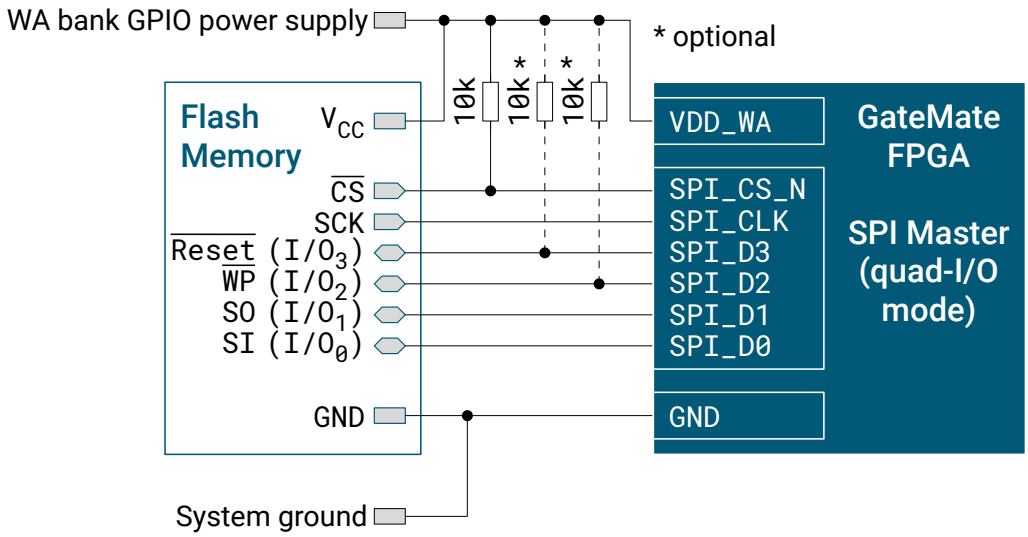


Figure 3.7: *Configuration data from flash memory in SPI quad-I/O mode*

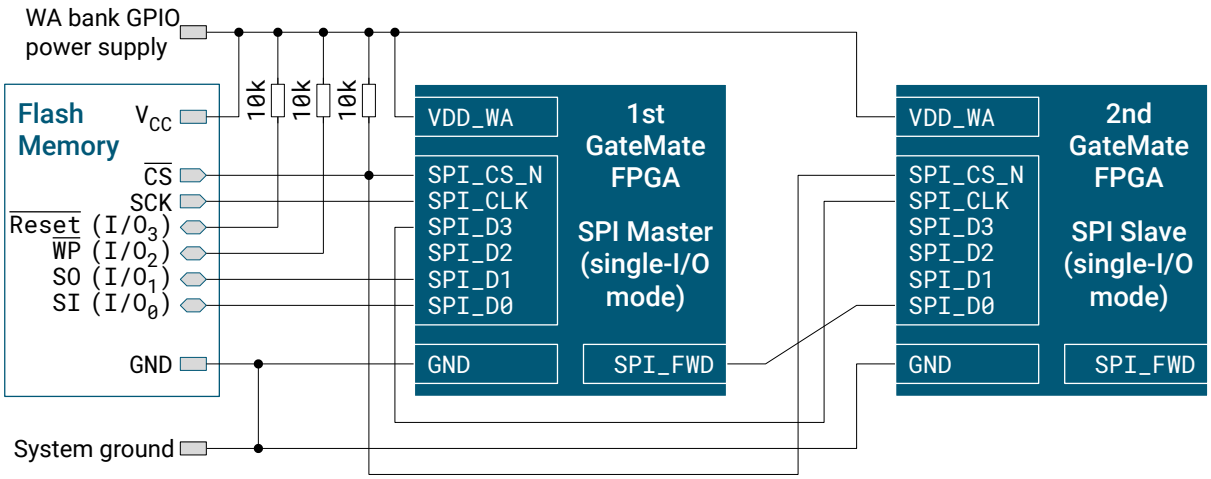


Figure 3.8: *Configuration data stream from a single flash memory*

Finally, Figure 3.8 shows two GateMate™ FPGAs using the same SPI configuration data source. This is done using the SPI forward function. Only SPI single-I/O mode can be used in this case. Furthermore, CPOL = 1 and CPHA = 1 must be set up (see configuration modes in Table 3.1 on page 140). In a multi-chip setup multiple FPGAs are configured in a chain.

The GateMate™ FPGA is able to load it's configuration automatically after reset. No external clock is required. When SPI active Mode is set up, the rising edge of RST_N starts reading the configuration data from the flash memory. The internal clock needs no external reference clock. If an external reference clock is available, the configuration stream can set up a phase-locked loop (PLL) to any configuration clock frequency.

When operating in SPI active mode, error detection and correction is done automatically. In this case a short CFG_FAILED_N pulse is given and the failed configuration sequence will be re-read.

When operating in SPI passive mode, the configuration controller can only react on the incoming data stream while the external device in SPI active mode is in control over the bus.

The signals CFG_DONE and CFG_FAILED_N indicate the end of the configuration process as described in Table 3.3.

Table 3.3: *Configuration process state*

CFG_DONE	CFG_FAILED_N	Description
0	0 *	Invalid state
0	1	Configuration procedure is still in progress
1	0 *	Unable to load configuration. Data stream might be invalid
1	1 **	Configuration successful

* Only long CFG_FAILED_N pulses. Short pulses have no significance.
** CFG_DONE might not be retrievable in case the configuration bank is set to user mode.

CFG_FAILED_N requires a pull-up resistor. If multiple FPGAs are configured from a single flash memory, all CFG_FAILED_N have to be connected as shown in Figure 3.9.

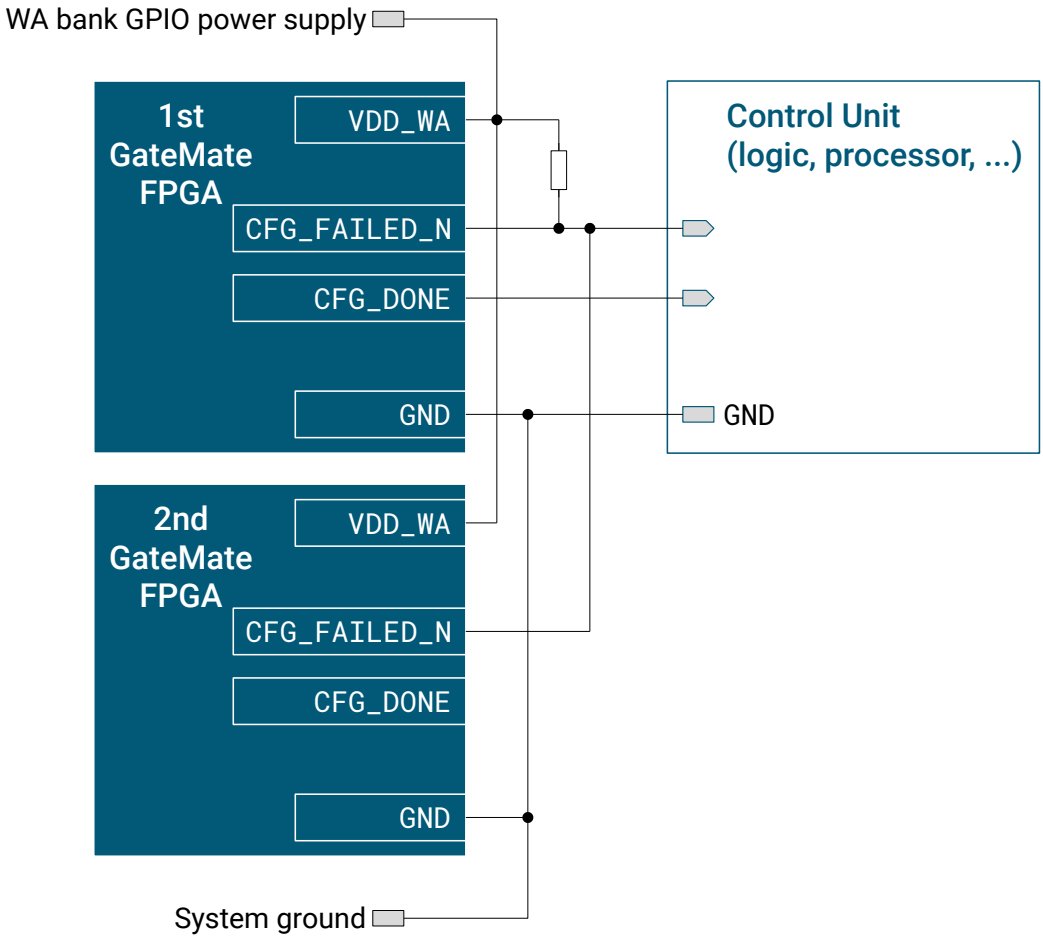


Figure 3.9: *Connection of CFG_DONE and CFG_FAILED_N signals in multi-FPGA applications with a single flash memory*

3.4 JTAG Interface

3.4.1 JTAG overview

The GateMate™ FPGA configuration bank provides a standard IEEE 1149.1-2001 compliant Test Access Point (TAP) that can be used for the following functions:

- Load configuration via JTAG [1]
- Full access to SPI interface using JTAG-SPI bypass
- Utilities to access external SPI data flashes
- Boundary-Scan Testability: `SAMPLE` / `PRELOAD` and `EXTEST`
- Access to supplementary internal registers

Figure 3.10 illustrates the typical wiring between a single FPGA and a JTAG host controller. By connecting devices via the data input and output signals it is possible to chain multiple TAPs in a serial configuration. The operating range of the supply voltage on the configuration bank is shown in Table 4.4 on page 161. The JTAG interface is always available after global reset. It will no longer be available once the configuration bank is switched to user I/O after configuration.

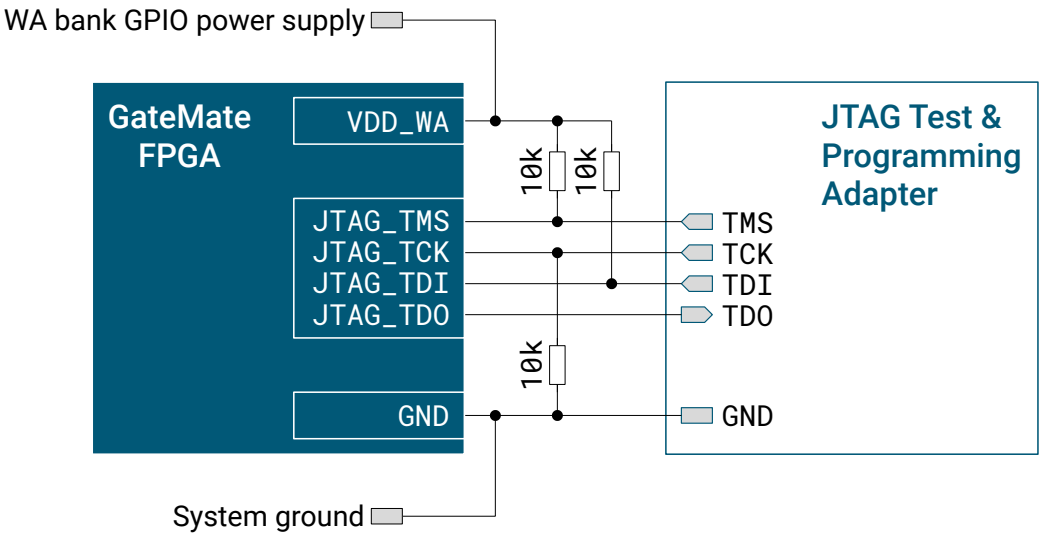


Figure 3.10: *JTAG connection scheme*

Four signals support the TAP according to the IEEE 1149.1-2001 requirements:

Test Mode Select (TMS) controls the TAP controller state machine. The signal is sampled on the rising edge of TCK.

Test Clock (TCK) provides a clock signal. TMS is sampled on the rising edge of TCK. Input data on TDI is shifted into the instruction (IR) or data (DR) registers on the rising edge of TCK. Output data on TDO is shifted out on the falling edge of TCK.

[1] See Section 3.3.1 for more ways to load the FPGA configuration.

Test Data Input (TDI) is the serial input to all JTAG instruction (IR) and data (DR) registers depending on the sequence previously applied at TMS.

Test Data Output (TDO) is the serial output for all JTAG instruction (IR) and data (DR) registers depending on the sequence previously applied at TMS.

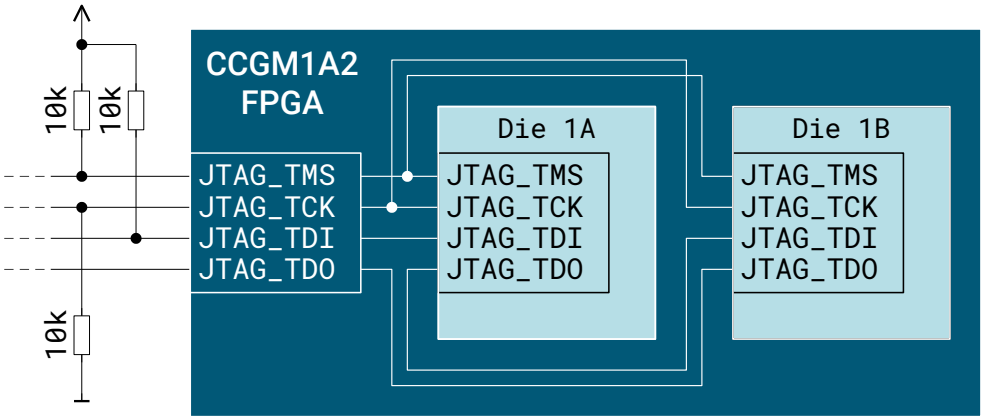


Figure 3.11: *Internal interposer circuit of the CCGM1A2 JTAG interface*

Within the CCGM1A2, the JTAG data ports of the two FPGA dies are connected in series as shown in Figure 3.11. In a JTAG device chain, the two dies of CCGM1A2 look like two individual CCGM1A1 devices.

CCGM1A2 can be identified via the code `0x009021`, which can be queried via JTAG as the status of general purpose input / output (GPIO) bank S1 of die 1A. This GPIO bank is not routed out to BGA balls on the CCGM1A2. The same query returns a different, arbitrary value for CCGM1A1, which depends on the user configuration, as this bank is of course routed to BGA balls and is therefore part of the user circuit.

3.4.2 JTAG Usage

The TMS signal is evaluated on the rising edge of TCK. Its value decides the next state into which the finite-state machine (FSM) changes. The FSM states are shown in Figure 3.12.

The typical procedure is to first load the instruction and then write or read the data associated with this instruction. The JTAG host controller must drive the signals TCK, TMS and TDI in the according way as illustrated shown in Figures 3.13 and 3.14. The instruction register (IR) size is 6 bits. Instructions have to be shifted into the JTAG TAP with least significant bit (LSB) first.

3.4.3 Common JTAG Instructions

According to IEEE 1149.1-2001, the `IDCODE` register is optional and thus only contains a placeholder value.

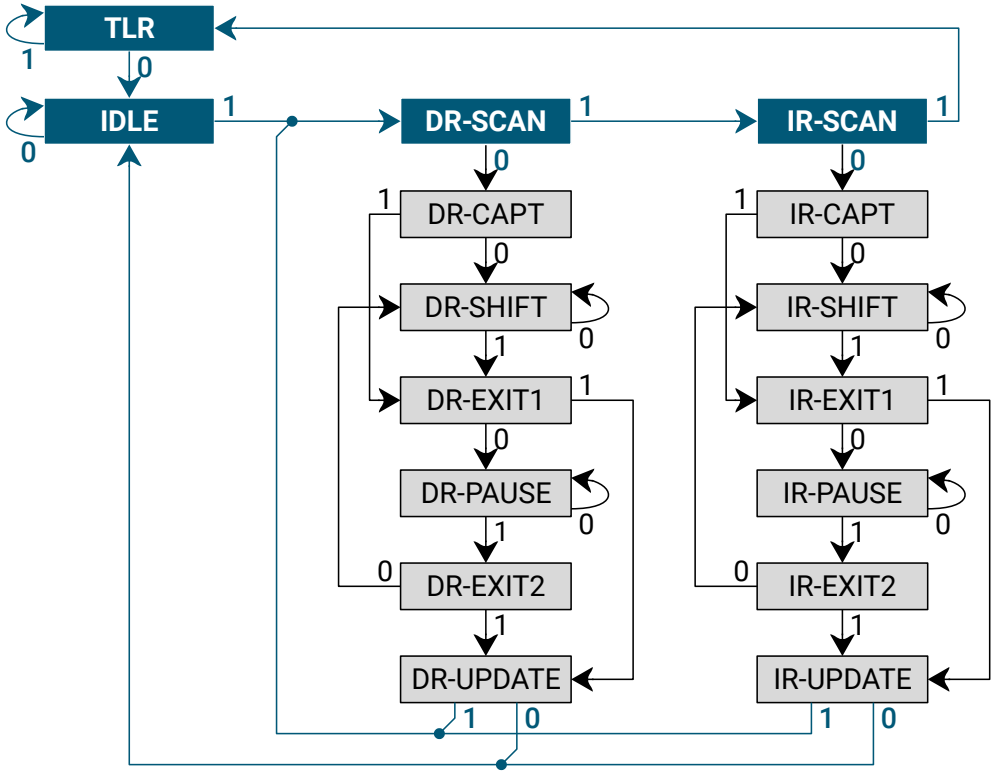


Figure 3.12: *Finite-state machine of the JTAG interface*

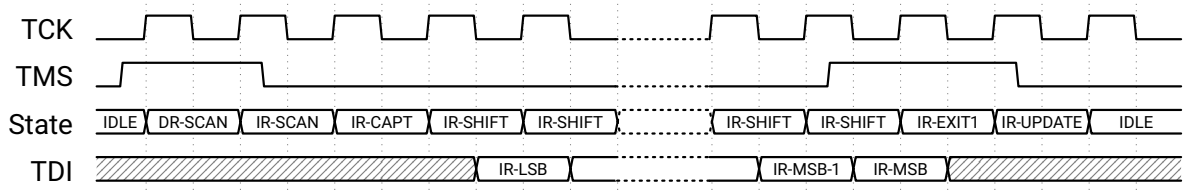


Figure 3.13: *JTAG instruction shift NEW COMMAND*

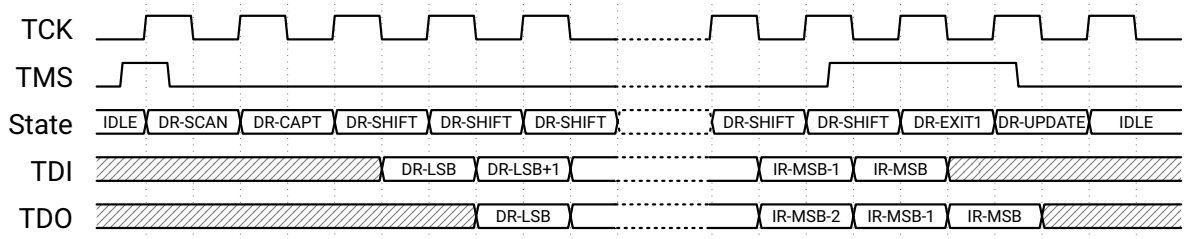


Figure 3.14: *JTAG data shift*

The bypass mode connects the JTAG TDI and TDO lines via a single-bit pass-through register and allows the testing of other devices in a JTAG chain. In bypass mode, the TDI input is delayed by one clock cycle before outputting to TDO.

Instruction:	0x 3F	JTAG_BYPASS	Shift width:	1 bit
		Configure JTAG bypass mode. The bypass register is set to zero when the TAP controller is in the CAPTURE-DR state.		
Shift-in bits:	-			
Shift-out bits:	-	Input bit pattern delayed by one clock cycle.		

3.4.4 JTAG Configuration

The JTAG interface can be used to configure the FPGA as well. The configuration stream must have a length of a multiple of 8 bits. The JTAG TAP automatically combines every 8 received bits to a single byte and forwards it to the configuration processing block. JTAG configuration works independently of the configuration mode pins CFG_MD[3:0]. However, in order to avoid conflicts, the pins should be set to 0x C. Further information on the configuration modes can be found in Section 3.3.1. The configuration byte stream is in big endian while the bytes are in little endian. If the configuration bank is not switched to user I/O after configuration, the status can be read as usual using the CFG_DONE and CFG_FAILED_N pins.

Instruction:	0x 06	JTAG_CONFIGURE	Shift width:	M · 8 bits
		Configure the FPGA with a configuration stream of M bytes.		
Shift-in bits:	M·[7:0]	The configuration byte stream is in big endian while the bytes are in little endian.		
Shift-out bits:	M·[7:0]	Don't care		

3.4.5 Access to SPI Bus

The SPI bus can be accessed directly using the JTAG interface. This way, bitstreams may be written to an external data flash without having to access the bus with additional hardware. There are two ways of acting on the SPI bus via the JTAG interface:

- Full access to SPI interface using JTAG-SPI bypass
- Utilities to access external SPI data flashes

Figure 3.15 illustrates the wiring between a single FPGA connected to an external SPI component and a JTAG host controller.

In active JTAG-SPI bypass mode all JTAG signals JTAG_TCK, JTAG_TDI and JTAG_TDO are directly connected to the SPI pins. Due to the direct connection of JTAG_TCK and

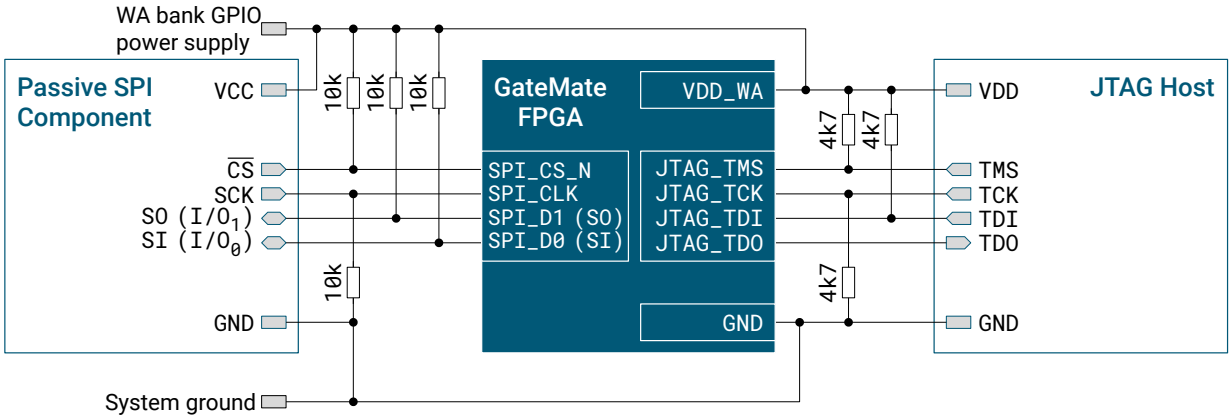


Figure 3.15: *JTAG-SPI bypass connection scheme*

SPI_CLK, the clock frequency on the SPI bus is the same as that of the JTAG clock. Before accessing the SPI bus via active bypass it must be ensured that the SPI bus is not busy, e.g., by checking the JTAG_GET_SPI_BUSY command. There is no restriction on the length of a SPI transfer. The SPI bypass mode works for all four SPI modes (CPOL, CPHA), but only in single-IO mode as set via the SPI configuration mode pins CFG_MD[1:0] described in Table 3.1 on page 140.

With each DR-shift one bit at JTAG_TDI is shifted into the JTAG TAP and directly forwarded to SPI_D0 output. Simultaneously, one bit is shifted into SPI_D1 input. The bit will be available on the JTAG output JTAG_TDO at the next falling edge of JTAG_TCK. There must be one more DR-shift cycle than bits to transmit. The last bit shifted into the TAP will not be shifted to the external SPI device and the very first bit shifted out of the TAP is an invalid bit.

Instruction:	**0x05**	**JTAG_SPI_BYPASS**	Shift width:	N bits
		Active access to SPI bus via JTAG.		
Shift-in bits:	-	Data to be shifted into external SPI device		
Shift-out bits:	-	Data to be shifted from external SPI device		

Figure 3.16 illustrates the function of the active JTAG-SPI bypass in all four available SPI single-IO modes.

3.4.6 Access to SPI data flash

The JTAG_SPI_ACCESS provides access to external SPI data flashes with a conventional command structure, in which command, mode, address and bidirectional data bits can be exchanged. In order to support a wide range of SPI data flashes, all command fields can be adjusted. First, the instructions have to be shifted into the JTAG TAP register with

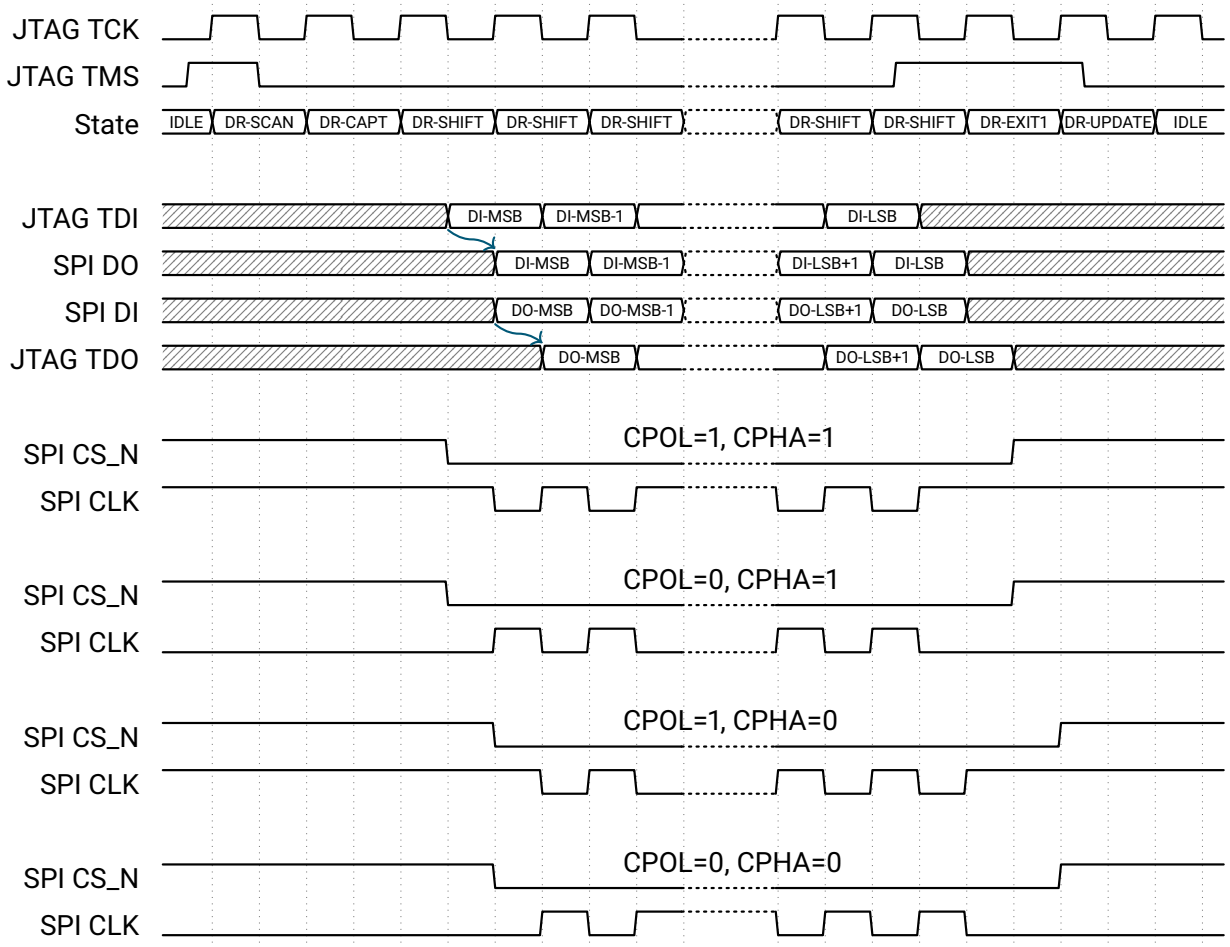


Figure 3.16: *Timing diagram of the JTAG-SPI bypass mode.*

LSB first. The actual SPI transfer starts as soon as the `DR-UPDATE` state in JTAG FSM was executed. In contrast to the JTAG-SPI bypass mode, the SPI clock rate is determined by the clock frequency of the internal oscillator. Access to external SPI data flashes works for all four SPI modes (CPOL, CPHA), but only in single-IO mode. The mode is again indicated by the configuration mode pins `CFG_MD[1:0]` as described in Table 3.1 on page 140. Setting `CFG_MD2` to one ensures the controller to be in SPI active mode.

The `JTAG_SPI_ACCESS` command allows transmissions of at most 32 TX data bits. The number of TX bits must be specified and TX words must be set to 1. Moreover, up to 32 bits of data can be received via the `JTAG_GET_SPI_RXDATA` command after execution of `JTAG_SPI_ACCESS` with the corresponding read command and number of bits to be received. The RX data flag is only valid if the read command was previously executed.

In order to transfer more than 32 bits, the JTAG TAP contains a register file of 256 bytes that can be filled via the JTAG command `JTAG_FLASH_PAGE_PREP` before the execution of an `JTAG_SPI_ACCESS` with a number of TX data word greater than 1. The total number of bits to be transmitted from the register file is the product of TX data bits times TX data words. The internal file register for transmitting more than 32 bits contains a 256 byte memory that can be pre-loaded by using the `JTAG_FLASH_PAGE_PREP` command.

An entire 32-bit word must always be followed by a DR-UPDATE. This procedure can be repeated up to 64 times to load a full page of 256 bytes. As long as no test logic reset command is executed, the file register retains its content, but can be overwritten. Its address counter is set to zero each time the command is called. For correct functioning it is necessary to transmit the first word of the register file again during JTAG_SPI_ACCESS.

It is highly recommended to verify the idle state before accessing the SPI bus by executing the JTAG_GET_SPI_BUSY command.

Instruction:	**0x01**	**JTAG_SPI_ACCESS**	Shift width:	119 bits
		Access external SPI data flash via JTAG.		
Shift-in bits:	[118:115]	Number of **command bits** to be shifted to SPI data flash		
	[114:109]	Number of **address bits** to be shifted to SPI data flash		
	[108:105]	Number of **mode bits** to be shifted to SPI data flash		
	[104: 99]	Number of **TX data bits** to be shifted to SPI data flash		
	[98: 92]	Number of **TX data words** to be shifted to SPI data flash		
	[91: 86]	Number of **dummy cycles** to be shifted to SPI data flash		
	[85: 80]	Number of **RX data bits** to be shifted to SPI data flash		
	[79: 72]	**Command** to be transferred to SPI data flash		
	[71: 64]	**Mode** to be transferred to SPI data flash		
	[63: 32]	**Address** to be transferred to SPI data flash		
	[31: 0]	**TX data** word to be transferred to SPI data flash. The byte stream is in big endian while the bytes are in little endian. Must contain the first word when transmitting data using the JTAG_FLASH_PAGE_PREP command.		
Shift-out bits:	[118: 0]	Don't care		

Instruction:	**0x02**	**JTAG_FLASH_PAGE_PREP**	Shift width:	$K \cdot 32$ bits
		Load data for writing up to one 256-byte flash page into FPGA. Must be followed by 0x1 – JTAG_SPI_ACCESS.		
Shift-in bits:	K·[31:0]	K 32-bit words to be written to one page in the SPI data flash with $1 \leqslant K \leqslant 64$. Each word must be followed by a DR-UPDATE. The byte stream is in big endian while the bytes are in little endian.		
Shift-out bits:	K·[31:0]	Don't care		

Instruction:	**0x03**	**JTAG_GET_SPI_RXDATA**	Shift width: 33 bits
		Get data read from SPI data flash via after 0x1 – JTAG_SPI_ACCESS.	
Shift-in bits:	[32:0]	Don't care	
Shift-out bits:	[32:1]	RX data read from SPI data flash	
	[0]	RX data valid flag. Only true if the preceding command was JTAG_SPI_ACCESS.	

Instruction:	**0x04**	**JTAG_GET_SPI_BUSY**	Shift width: 1 bit
		Checks if the SPI controller is busy. Recommended to use before any SPI access.	
Shift-in bits:	[0:0]	Don't care	
Shift-out bits:	[0:0]	Obtain SPI bus state: 0b0: SPI idle 0b1: SPI busy	

3.4.7 SerDes register file Interface

The serializer / deserializer (SerDes) register file can be accessed via the FPGA fabric as well as the JTAG interface with priority given to the JTAG interface if access is simultaneous. For more details see Section 2.5 from page 50.

Instruction:	**0x25**	**JTAG_WR_SERDES_REGFILE**	Shift width: 41 bits
		Reads and writes SerDes register file. By not setting write-enable no data is written and there is just data read. Has to be followed by instruction 0x26 to get last read data in any case.	
Shift-in bits:	[40]	Write-enable bit	
	[39:24]	Bitmask for writing data	
	[23:8]	Data to write	
	[7:0]	Adress	
Shift-out bits:	[40:0]	Don't care	

Instruction:	0x26	JTAG_RD_SERDES_REGFILE	Shift width:	16 bits
		Gets last data read from SerDes register file		
Shift-in bits:	[15:0]	Don't care		
Shift-out bits:	[15:0]	Data read by last command: 0x25 - Read/write SerDes register file		

3.4.8 Boundary-Scan Test

A JTAG IEEE 1149.1-2001 compliant boundary-scan allows to test pin connections without physically probing pins by shifting a boundary-scan register that is composed of cells connected between the FPGA logic and IOs.

The SAMPLE instruction allows to take a snapshot of the input and output signal states without interfering with the device pins. The snapshot is taken on the rising edge of JTAG_TCK in the DR-CAPT state. The data can then be accessed by shifting through the JTAG_TDO output.

The PRELOAD instruction allows scanning of the boundary-scan register without causing interference to the FPGA logic. It allows an initial pattern to be shifted into the boundary-scan register cells.

Both instructions SAMPLE and PRELOAD are combined in the JTAG_SAMPLE_PRELOAD command. While preloading the boundary-scan register cells by shifting data into the input JTAG_TDI the sample results are shifted through the JTAG_TDO output.

The EXTEST instruction allows access to the external circuitry through the device pins. The boundary-scan register cells connected to the output pins are used to apply test stimuli, while those at the input pins capture external signals from the device pins.

The boundary-scan register length is 317 bits. To enable boundary-scan test, the test-mode register must be set first by using the JTAG_SET_TESTMODE instruction. Please refer to the official Boundary Scan Description Language (BSDL) files for more information.

Instruction:	0x20	JTAG_SET_TESTMODE	Shift width:	2 bits
		Set value in testmode register.		
Shift-in bits:	[1:0]	Testmode to be set: 0b 00: Disable testmode 0b 01: reserved 0b 10: reserved 0b 11: Boundary-Scan Test		
Shift-out bits:	[1:0]	Don't care		

Instruction:	**0x 3D**	**JTAG_SAMPLE_PRELOAD**	Shift width:	317 bits
		Boundary-scan SAMPLE and PRELOAD instruction. The register 0x 20 – Set Testmode must be set to 0b 11.		
Shift-in bits:	[316:0]	Preload value for boundary-scan shift register.		
Shift-out bits:	[316:0]	Sample value of boundary-scan shift register.		

Instruction:	**0x 3E**	**JTAG_EXTEST**	Shift width:	317 bits
		Boundary-scan EXTEST instruction. The register 0x 20 – Set Testmode must be set to 0b 11.		
Shift-in bits:	[316:0]	Don't care.		
Shift-out bits:	[316:0]	Value of boundary-scan shift register.		

Chapter 4

Electrical Characteristics

Table 4.1: *Absolute maximum ratings*

Symbol	Min	Typ	Max	Unit	Description
T_J	−40		125	°C	Junction temperature
VDD_{IO}			2.75	V	I/O supply voltage (GPIO banks and VDD_CLK)
VDD_{core}			1.20	V	Core supply voltage (VDD, VDD_PLL, VDD_SER and VDD_SER_PLL)
V_{ESD}			4000	V	HBM ESD class 2

Table 4.2: *Operation range*

Symbol	Min	Typ	Max	Unit	Description
T_J	−40		125	°C	Junction operating temperature
$VDD_{low\,power}$	0.85	0.9	0.95	V	Core supply voltage (VDD, VDD_PLL) in low power mode
$VDD_{economy}$	0.95	1.0	1.05	V	Core supply voltage (VDD, VDD_PLL) in economy mode
VDD_{speed}	1.05	1.1	1.15	V	Core supply voltage (VDD, VDD_PLL) in speed mode
I_{leak,low_power}					CCGM1A1 leakage current in low power mode
		2		mA	at −20 °C
		8		mA	at +20 °C
		96		mA	at +85 °C
$I_{leak,economy}$					CCGM1A1 leakage current in economy mode
		3		mA	at −20 °C
		12		mA	at +20 °C
		130		mA	at +85 °C
$I_{leak,speed}$					CCGM1A1 leakage current in speed mode
		5		mA	at −20 °C
		19		mA	at +20 °C
		170		mA	at +85 °C
VDD_{SER}	0.95		1.15	V	SerDes supply voltage (VDD_SER)
VDD_{SER_PLL}	0.95		1.15	V	SerDes PLL supply voltage (VDD_SER_PLL)

Table 4.3: *Serializer / deserializer (SerDes) electrical characteristics*

Symbol	Min	Typ	Max	Unit	Description
T_{SER}	−40	25	125	°C	SerDes operating temperature
VDD_{SER}	0.95	1.0	1.1	V	VDD_SER supply voltage range
VDD_{SER_PLL}	0.95	1.0	1.1	V	VDD_SER_PLL supply voltage range
f_{SER}		100	125	MHz	SER_CLK reference clock frequency
J_{SER}			1	ps	SER_CLK reference clock jitter
$t_{dly,lowpower}$	50	65	85	ps	Step size of the programmable delay element, 15 levels adjustable
$t_{dly,economy}$	38	50	65	ps	dto.
$t_{dly,speed}$	30	38	50	ps	dto.
$R_{TERM,REF}$	198	200	202	Ω	Internal calibration reference resistor
R_{TERM}		100		Ω	Internal differential termination impedance
D_{SER}	0.3		5	Gbit/s	Data rate
f_{DCO}	1250		2500	MHz	Internal frequency range of the SerDes PLL

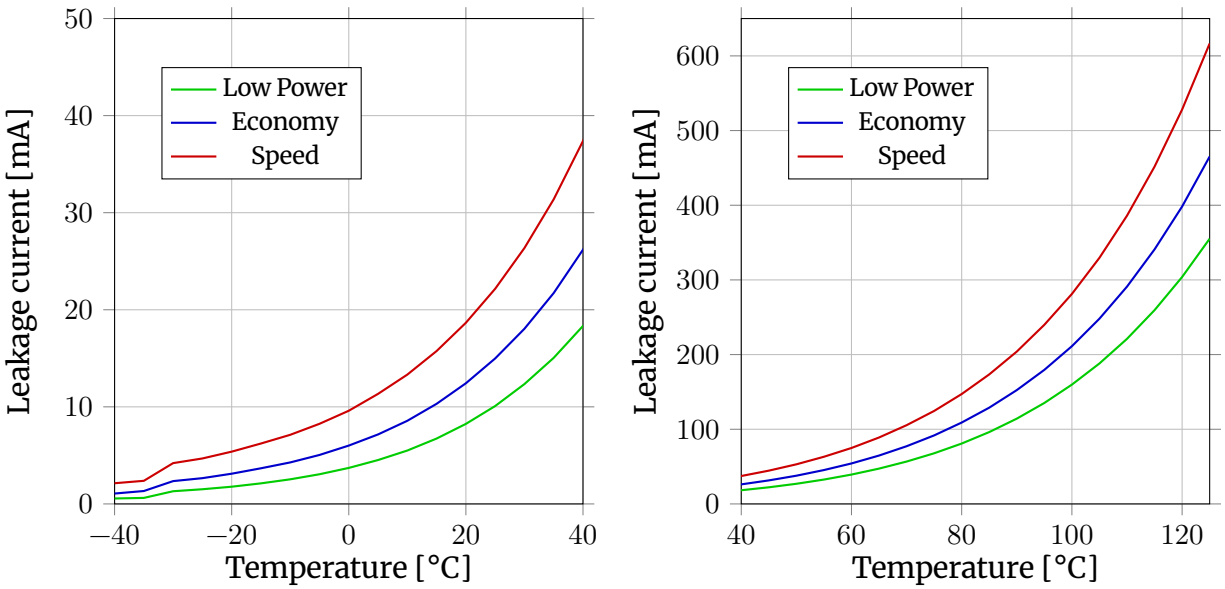


Figure 4.1: *CCGM1A1 leakage current*

Figures 4.2 and 4.3 are based on a stress test where a maximum FPGA load is generated. Most available logic resources (LUTs & flip-flops) are activated. The design implements a modified Galois linear feedback shift register (Galois-LFSR), generating a lot of "chaotic" switching activity. For further information, please see FPGA Stress Test.

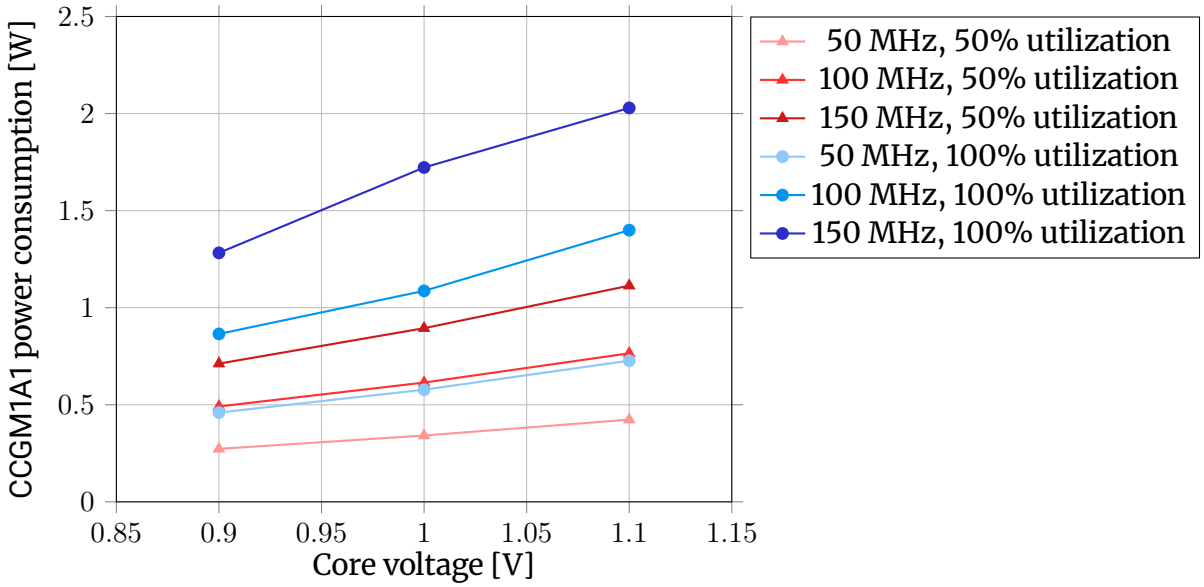


Figure 4.2: *CCGM1A1 total power consumption based on Galois-LFSR*

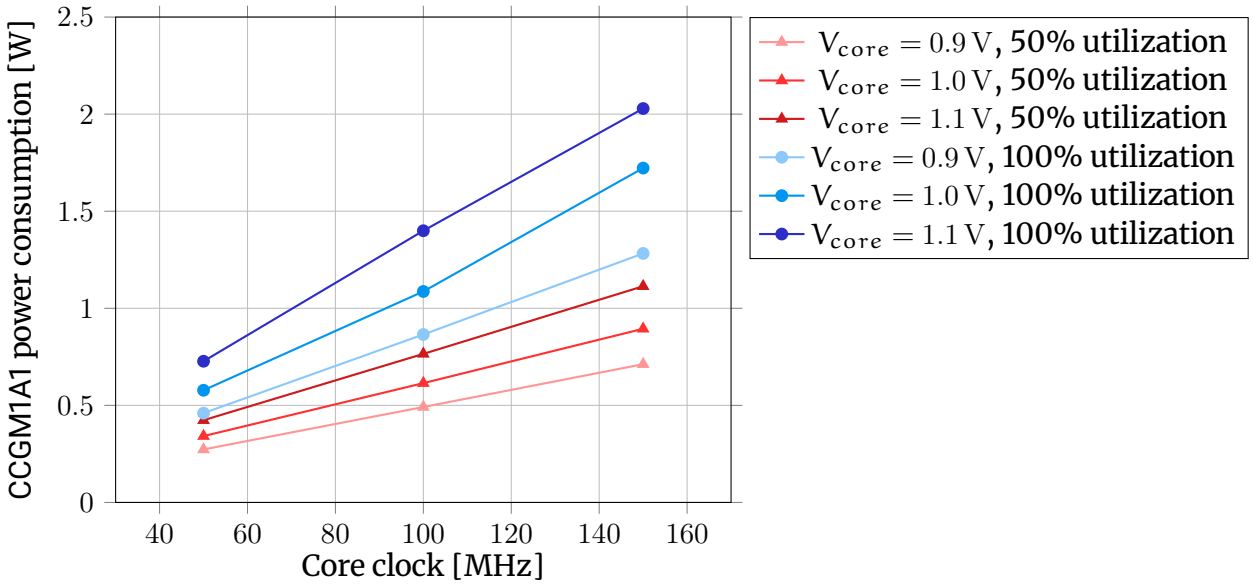


Figure 4.3: *CCGM1A1 total power consumption based on Galois-LFSR*

Table 4.4: *GPIO characteristics in single-ended mode*

Symbol	Min	Typ	Max	Unit	Description
VDD_{IO}	1.1		2.7	V	I/O supply voltage (GPIO banks and VDD_CLK)
V_{IN}	−0.4		$VDD_{IO} + 0.4$	V	Input voltage range
Schmitt-trigger function disabled:					
V_{IH}	0.43		0.51	VDD_{IO}	Input high threshold voltage
V_{IL}	0.45		0.51	VDD_{IO}	Input low threshold voltage
V_{HYST}		0		V	Hysteresis
Schmitt-trigger function enabled:					
V_{IH}	0.61		0.67	VDD_{IO}	Input high threshold voltage
V_{IL}	0.31		0.39	VDD_{IO}	Input low threshold voltage
V_{HYST}	0.26		0.33	VDD_{IO}	Hysteresis
$t_{dly,lowpower}$	50	65	85	ps	Step size of the programmable delay element, 15 levels adjustable
$t_{dly,economy}$	38	50	65	ps	dto.
$t_{dly,speed}$	30	38	50	ps	dto.
Output driver disabled:					
I_{IL}			1	µA	Input pin current when driven active low
I_{IH}			1	µA	Input pin current when driven active high
R_{PU}		50		kΩ	Pull-up resistance
R_{PD}		50		kΩ	Pull-down resistance
$I_{DD,max}$			158	µA	Maximum supply current at GPIO input at transition point

Table 4.5: *GPIO characteristics in LVDS mode*

Symbol	Min	Typ	Max	Unit	Description
VDD_{IO}	1.62		2.75	V	I/O supply voltage
I_{out}		3.2		mA	Output current when LVDS output current boost is set to nominal current
I_{out}		6.4		mA	Output current when LVDS output current boost is set to increased output current
VCM_{TX}		$VDD_{IO}/2$		V	Common-mode voltage
R_{term}	90	100	130	Ω	

Table 4.6: *PLL characteristics*

Symbol	Min	Typ	Max	Unit	Description
$VDD_{PLL,low\,power}$	0.85	0.9	0.95	V	Clean analog supply voltage for DCO in low power mode
$VDD_{PLL,economy}$	0.95	1.0	1.05	V	Clean analog supply voltage for DCO in economy mode
$VDD_{PLL,speed}$	1.05	1.1	1.15	V	Clean analog supply voltage for DCO in speed mode
$VDD_{CLK,low\,power}$	0.85	0.9	0.95	V	Clock generator digital supply voltage in low power mode
$VDD_{CLK,economy}$	0.95	1.0	1.05	V	Clock generator digital supply voltage in economy mode
$VDD_{CLK,speed}$	1.05	1.1	1.15	V	Clock generator digital supply voltage in speed mode
t_{lock}			1100	Number of ref. clock cycles	Lock in time (coarse tune, fast-lock mode) with lock detection
t_{lock}			57	Number of ref. clock cycles	Lock in time (coarse tune, fast-lock mode) without lock detection
$f_{DCO,\,low\,power}$	500		1000	MHz	DCO frequency in low power mode
$f_{DCO,\,economy}$	1000		2000	MHz	DCO frequency in economy mode
$f_{DCO,\,speed}$	1250		2500	MHz	DCO frequency in speed mode
$T_{DCO,\,step}$			10	ps	DCO period tuning step size

Table 4.7: *Reset characteristics*

Symbol	Min	Typ	Max	Unit	Description
V_{DD}	0.85	1.0	1.15	V	Core supply voltage
$V_{core,TH+}$	0.62	0.72	0.80	V	High reset threshold voltage
$V_{core,TH-}$	0.56	0.62	0.71	V	Low reset threshold voltage
$V_{core,hyst}$	60	82	96	mV	Hysteresis voltage
SR_{core}	1.8		180	V/ms	Supply voltage slew rate
VDD_{CLK}	$1.80 \pm 10\%$		$2.50 \pm 10\%$	V	VDD_CLK supply voltage
=	1.62		2.75	V	
$V_{IO,TH+}$	1.21	1.44	1.56	V	High reset threshold voltage [1]
	1.01	1.18	1.34	V	ditto [2]
	0.95	1.08	1.20	V	ditto [3]
$V_{IO,TH-}$	1.00	1.20	1.35	V	Low reset threshold voltage [1]
	0.89	1.06	1.19	V	ditto [2]
	0.88	0.98	1.05	V	ditto [3]
$V_{IO,hyst}$	152	180	215	mV	Hysteresis voltage [1]
	108	126	177	mV	ditto [2]
	85	100	110	mV	ditto [3]
SR_{IO}	1.8		180	V/ms	Supply voltage slew rate

[1] No level adjustment, POR_ADJ open at CCGM1A1 (pin is not available at CCGM1A2).
[2] Level adjustment with R3a $= 500\,k\Omega$ (see Figure 3.4 on page 138).
[3] Level adjustment with R3a $= 250\,k\Omega$.

Chapter 5

CCGM1A1 / CCGM1A2 Pinout

5.1 Overview

The GateMate™ FPGAs CCGM1A1 / CCGM1A2 have a 18×18 pin 0.8 mm fine pitch Ball Grid Array (FBGA) package with 324 balls.

General purpose input / output (GPIO) ball arrangement is defined for 4-layer printed circuit board (PCB) layout with only 2 signal layers.

CCGM1A1 and CCGM1A2 are pin compatible, which is why the same design can be used for both FPGAs if the PCB circuitry is designed accordingly.

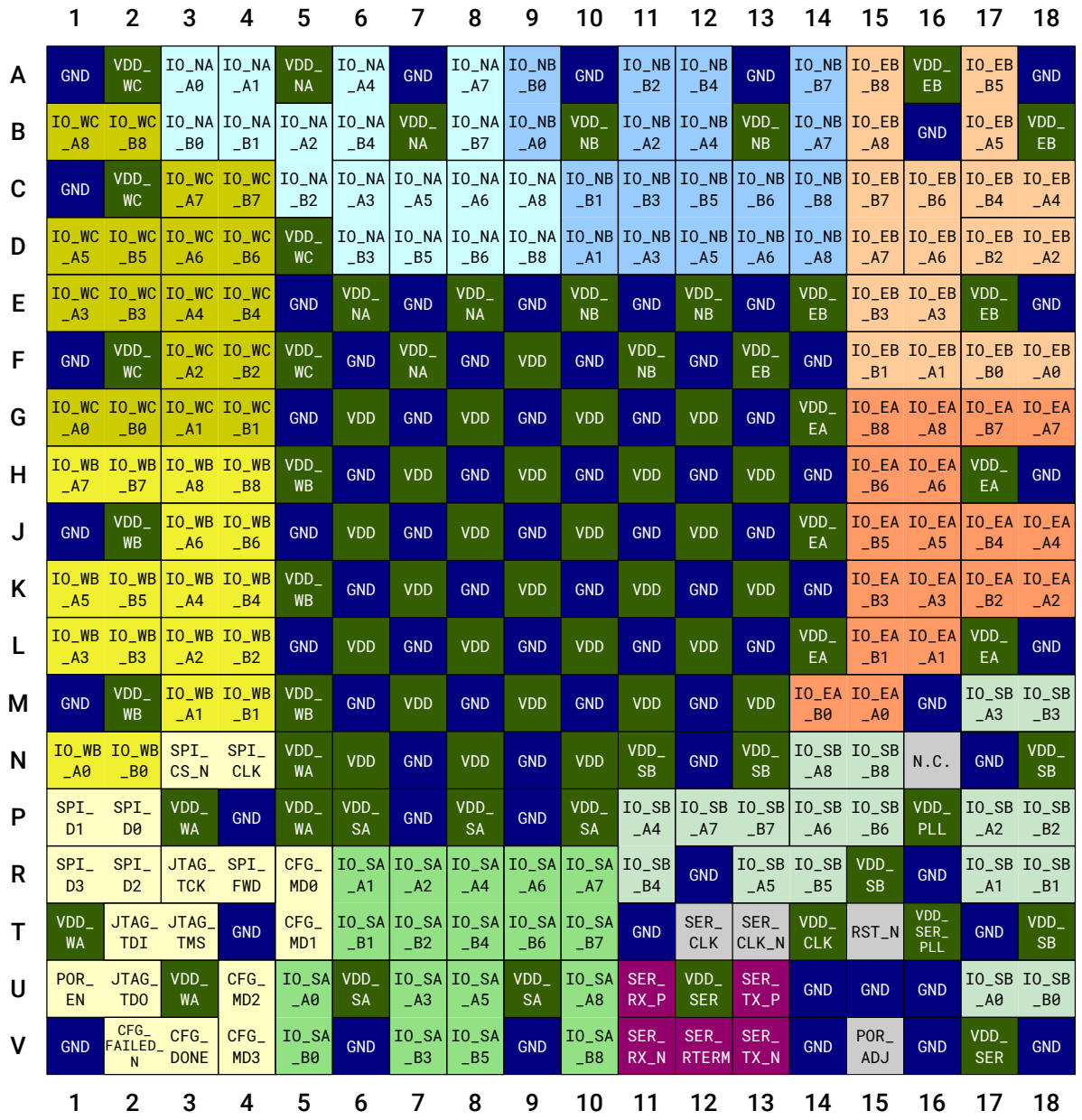
CCGM1A2 consists of two dies with CCGM1A1 functionality. Since they both have the same package, some GPIO banks are not accessible via BGA balls. Figure 2.8 on page 30 shows how the connected GPIO banks are allocated to the dies. A brief overview is shown in Table 5.1.

Table 5.1: *GPIO bank allocation of CCGM1A2*

BGA bank	from die 1A	from die 1B
IO_SA_[A|B][8:0]	SA	
IO_SB_[A|B][8:0]	SB	SB
IO_WA_[A|B][8:0]	WA	
IO_WB_[A|B][8:0]	EA	WB
IO_WC_[A|B][8:0]	WC	
IO_NA_[A|B][8:0]	NA	NA
IO_NB_[A|B][8:0]	NB	
IO_EA_[A|B][8:0]		EB
IO_EB_[A|B][8:0]		EB

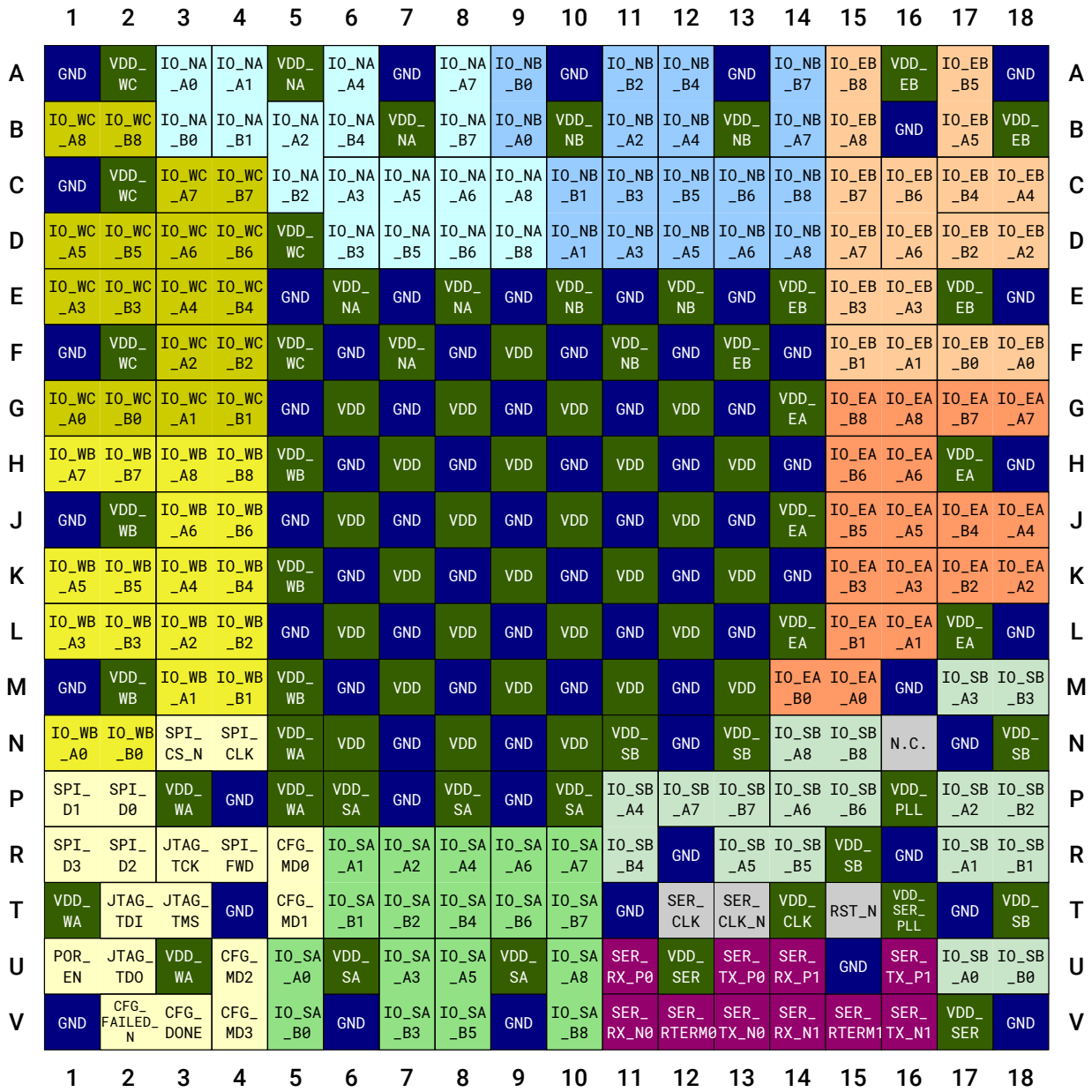
Figures 5.1 and 5.2 show the pinout of either FPGAs. a detailed pin list of the chips is given in Section 5.2 from page 170 and Section 5.3 from page 181.

There are only a few pins where the two chips are different. These pins are shown in Figure 5.3. PCBs designed for both FPGAs should take these differences into account.



	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	
A	GND	VDD_WC	IO_NA_A0	IO_NA_A1	VDD_NA	IO_NA_A4	GND	IO_NA_A7	IO_NB_B0	GND	IO_NB_B2	IO_NB_B4	GND	IO_NB_B7	IO_EB_B8	VDD_EB	IO_EB_B5	GND	A
B	IO_WC_A8	IO_WC_B8	IO_NA_B0	IO_NA_B1	IO_NA_A2	IO_NA_B4	VDD_NA	IO_NA_B7	IO_NB_A0	VDD_NB	IO_NB_A2	IO_NB_A4	VDD_NB	IO_NB_A7	IO_EB_A8	GND	IO_EB_A5	VDD_EB	B
C	GND	VDD_WC	IO_WC_A7	IO_WC_B7	IO_NA_B2	IO_NA_A3	IO_NA_A5	IO_NA_A6	IO_NA_A8	IO_NB_B1	IO_NB_B3	IO_NB_B5	IO_NB_B6	IO_NB_B8	IO_EB_B7	IO_EB_B6	IO_EB_B4	IO_EB_A4	C
D	IO_WC_A5	IO_WC_B5	IO_WC_A6	IO_WC_B6	VDD_WC	IO_NA_B3	IO_NA_B5	IO_NA_B6	IO_NA_B8	IO_NB_A1	IO_NB_A3	IO_NB_A5	IO_NB_A6	IO_NB_A8	IO_EB_A7	IO_EB_A6	IO_EB_B2	IO_EB_A2	D
E	IO_WC_A3	IO_WC_B3	IO_WC_A4	IO_WC_B4	GND	VDD_NA	GND	VDD_NA	GND	VDD_NB	GND	VDD_NB	GND	VDD_EB	IO_EB_B3	IO_EB_A3	VDD_EB	GND	E
F	GND	VDD_WC	IO_WC_A2	IO_WC_B2	VDD_WC	GND	VDD_NA	GND	VDD	GND	VDD_NB	GND	VDD_EB	GND	IO_EB_B1	IO_EB_A1	IO_EB_B0	IO_EB_A0	F
G	IO_WC_A0	IO_WC_B0	IO_WC_A1	IO_WC_B1	GND	VDD	GND	VDD	GND	VDD	GND	VDD	GND	VDD_EA	IO_EA_B8	IO_EA_A8	IO_EA_B7	IO_EA_A7	G
H	IO_WB_A7	IO_WB_B7	IO_WB_A8	IO_WB_B8	VDD_WB	GND	VDD	GND	VDD	GND	VDD	GND	VDD	GND	IO_EA_B6	IO_EA_A6	VDD_EA	GND	H
J	GND	VDD_WB	IO_WB_A6	IO_WB_B6	GND	VDD	GND	VDD	GND	VDD	GND	VDD	GND	VDD_EA	IO_EA_B5	IO_EA_A5	IO_EA_B4	IO_EA_A4	J
K	IO_WB_A5	IO_WB_B5	IO_WB_A4	IO_WB_B4	VDD_WB	GND	VDD	GND	VDD	GND	VDD	GND	VDD	GND	IO_EA_B3	IO_EA_A3	IO_EA_B2	IO_EA_A2	K
L	IO_WB_A3	IO_WB_B3	IO_WB_A2	IO_WB_B2	GND	VDD	GND	VDD	GND	VDD	GND	VDD	GND	VDD_EA	IO_EA_B1	IO_EA_A1	VDD_EA	GND	L
M	GND	VDD_WB	IO_WB_A1	IO_WB_B1	VDD_WB	GND	VDD	GND	VDD	GND	VDD	GND	VDD	IO_EA_B0	IO_EA_A0	GND	IO_SB_A3	IO_SB_B3	M
N	IO_WB_A0	IO_WB_B0	SPI_CS_N	SPI_CLK	VDD_WA	VDD	GND	VDD	GND	VDD	VDD_SB	GND	VDD_SB	IO_SB_A8	IO_SB_B8	N.C.	GND	VDD_SB	N
P	SPI_D1	SPI_D0	VDD_WA	GND	VDD_WA	VDD_SA	GND	VDD_SA	GND	VDD_SA	IO_SB_A4	IO_SB_A7	IO_SB_B7	IO_SB_A6	IO_SB_B6	VDD_PLL	IO_SB_A2	IO_SB_B2	P
R	SPI_D3	SPI_D2	JTAG_TCK	SPI_FWD	CFG_MD0	IO_SA_A1	IO_SA_A2	IO_SA_A4	IO_SA_A6	IO_SA_A7	IO_SB_B4	GND	IO_SB_A5	IO_SB_B5	VDD_SB	GND	IO_SB_A1	IO_SB_B1	R
T	VDD_WA	JTAG_TDI	JTAG_TMS	GND	CFG_MD1	IO_SA_B1	IO_SA_B2	IO_SA_B4	IO_SA_B6	IO_SA_B7	GND	SER_CLK	SER_CLK_N	VDD_CLK	RST_N	VDD_SER_PLL	GND	VDD_SB	T
U	POR_EN	JTAG_TDO	VDD_WA	CFG_MD2	IO_SA_A0	VDD_SA	IO_SA_A3	IO_SA_A5	VDD_SA	IO_SA_A8	SER_RX_P	VDD_SER	SER_TX_P	GND	GND	GND	IO_SB_A0	IO_SB_B0	U
V	GND	CFG_FAILED_N	CFG_DONE	CFG_MD3	IO_SA_B0	GND	IO_SA_B3	IO_SA_B5	GND	IO_SA_B8	SER_RX_N	SER_RTERM	SER_TX_N	GND	POR_ADJ	GND	VDD_SER	GND	V
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	

Figure 5.1: *CCGM1A1 FBGA pinout*



	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	
A	GND	VDD_WC	IO_NA_A0	IO_NA_A1	VDD_NA	IO_NA_A4	GND	IO_NA_A7	IO_NB_B0	GND	IO_NB_B2	IO_NB_B4	GND	IO_NB_B7	IO_EB_B8	VDD_EB	IO_EB_B5	GND	A
B	IO_WC_A8	IO_WC_B8	IO_NA_B0	IO_NA_B1	IO_NA_A2	IO_NA_B4	VDD_NA	IO_NA_B7	IO_NB_A0	VDD_NB	IO_NB_A2	IO_NB_A4	VDD_NB	IO_NB_A7	IO_EB_A8	GND	IO_EB_A5	VDD_EB	B
C	GND	VDD_WC	IO_WC_A7	IO_WC_B7	IO_NA_B2	IO_NA_A3	IO_NA_A5	IO_NA_A6	IO_NA_A8	IO_NB_B1	IO_NB_B3	IO_NB_B5	IO_NB_B6	IO_NB_B8	IO_EB_B7	IO_EB_B6	IO_EB_B4	IO_EB_A4	C
D	IO_WC_A5	IO_WC_B5	IO_WC_A6	IO_WC_B6	VDD_WC	IO_NA_B3	IO_NA_B5	IO_NA_B6	IO_NA_B8	IO_NB_A1	IO_NB_A3	IO_NB_A5	IO_NB_A6	IO_NB_A8	IO_EB_A7	IO_EB_A6	IO_EB_B2	IO_EB_A2	D
E	IO_WC_A3	IO_WC_B3	IO_WC_A4	IO_WC_B4	GND	VDD_NA	GND	VDD_NA	GND	VDD_NB	GND	VDD_NB	GND	VDD_EB	IO_EB_B3	IO_EB_A3	VDD_EB	GND	E
F	GND	VDD_WC	IO_WC_A2	IO_WC_B2	VDD_WC	GND	VDD_NA	GND	VDD	GND	VDD_NB	GND	VDD_EB	GND	IO_EB_B1	IO_EB_A1	IO_EB_B0	IO_EB_A0	F
G	IO_WC_A0	IO_WC_B0	IO_WC_A1	IO_WC_B1	GND	VDD	GND	VDD	GND	VDD	GND	VDD	GND	VDD_EA	IO_EA_B8	IO_EA_A8	IO_EA_B7	IO_EA_A7	G
H	IO_WB_A7	IO_WB_B7	IO_WB_A8	IO_WB_B8	VDD_WB	GND	VDD	GND	VDD	GND	VDD	GND	VDD	GND	IO_EA_B6	IO_EA_A6	VDD_EA	GND	H
J	GND	VDD_WB	IO_WB_A6	IO_WB_B6	GND	VDD	GND	VDD	GND	VDD	GND	VDD	GND	VDD_EA	IO_EA_B5	IO_EA_A5	IO_EA_B4	IO_EA_A4	J
K	IO_WB_A5	IO_WB_B5	IO_WB_A4	IO_WB_B4	VDD_WB	GND	VDD	GND	VDD	GND	VDD	GND	VDD	GND	IO_EA_B3	IO_EA_A3	IO_EA_B2	IO_EA_A2	K
L	IO_WB_A3	IO_WB_B3	IO_WB_A2	IO_WB_B2	GND	VDD	GND	VDD	GND	VDD	GND	VDD	GND	VDD_EA	IO_EA_B1	IO_EA_A1	VDD_EA	GND	L
M	GND	VDD_WB	IO_WB_A1	IO_WB_B1	VDD_WB	GND	VDD	GND	VDD	GND	VDD	GND	VDD	IO_EA_B0	IO_EA_A0	GND	IO_SB_A3	IO_SB_B3	M
N	IO_WB_A0	IO_WB_B0	SPI_CS_N	SPI_CLK	VDD_WA	VDD	GND	VDD	GND	VDD	VDD_SB	GND	VDD_SB	IO_SB_A8	IO_SB_B8	N.C.	GND	VDD_SB	N
P	SPI_D1	SPI_D0	VDD_WA	GND	VDD_WA	VDD_SA	GND	VDD_SA	GND	VDD_SA	IO_SB_A4	IO_SB_A7	IO_SB_B7	IO_SB_A6	IO_SB_B6	VDD_PLL	IO_SB_A2	IO_SB_B2	P
R	SPI_D3	SPI_D2	JTAG_TCK	SPI_FWD	CFG_MD0	IO_SA_A1	IO_SA_A2	IO_SA_A4	IO_SA_A6	IO_SA_A7	IO_SB_B4	GND	IO_SB_A5	IO_SB_B5	VDD_SB	GND	IO_SB_A1	IO_SB_B1	R
T	VDD_WA	JTAG_TDI	JTAG_TMS	GND	CFG_MD1	IO_SA_B1	IO_SA_B2	IO_SA_B4	IO_SA_B6	IO_SA_B7	GND	SER_CLK	SER_CLK_N	VDD_CLK	RST_N	VDD_SER_PLL	GND	VDD_SB	T
U	POR_EN	JTAG_TDO	VDD_WA	CFG_MD2	IO_SA_A0	VDD_SA	IO_SA_A3	IO_SA_A5	VDD_SA	IO_SA_A8	SER_RX_P0	VDD_SER	SER_TX_P0	SER_RX_P1	GND	SER_TX_P1	IO_SB_A0	IO_SB_B0	U
V	GND	CFG_FAILED_N	CFG_DONE	CFG_MD3	IO_SA_B0	GND	IO_SA_B3	IO_SA_B5	GND	IO_SA_B8	SER_RX_N0	SER_RTERM0	SER_TX_N0	SER_RX_N1	SER_RTERM1	SER_TX_N1	VDD_SER	GND	V
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	

Figure 5.2: *CCGM1A2 FBGA pinout*

Figure 5.3: *Differences between CCGM1A1 and CCGM1A2 pinout*

Some pins have a second function as shown in Figures 5.4 and 5.5.

In both cases, one pin function is GPIO, and according to Figure 5.4, the configuration bank can alternatively be used as GPIO bank WA, and according to Figure 5.5, some GPIO pins can be used as dedicated clock inputs.

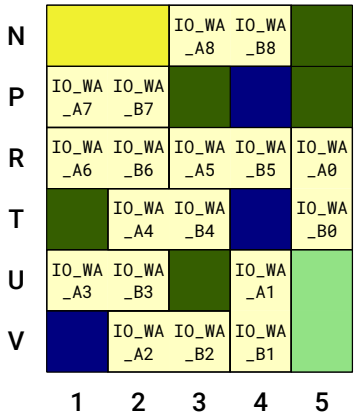


Figure 5.4: *Configuration pins of CCGM1A1/CCGM1A2 pinout for use as GPIO after configuration*

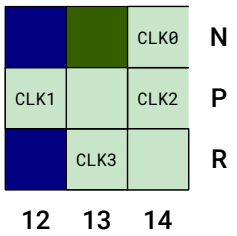


Figure 5.5: *Clock input pins as second function of some GPIOs*

All ball signals are grouped and color-coded as shown in Table 5.2.

Besides the 9 GPIO banks there are pin groups for the serializer / deserializer (SerDes) interface, power supply, ground and a few pins in the gray marked group for other signals.

The pin lists in Sections 5.2 and 5.3 below are also available for download in csv[1] file format.

[1] csv: comma-separated values

Table 5.2: *Pin types*

Color	Pin type	Number of pins
	North GPIO banks NA and NB Each bank has 18 GPIO signals and both have separate power supply balls and can be driven individually. GPIO signals are grouped by pairs and each pair can get configured to be either two single ended or a differential pair signal.	36
	East GPIO banks EA and EB Each bank has 18 GPIO signals and both have separate power supply balls and can be driven individually. GPIO signals are grouped by pairs and each pair can get configured to be either two single ended or a differential pair signal.	36
	South GPIO banks SA and SB Each bank has 18 GPIO signals and both have separate power supply balls and can be driven individually. GPIO signals are grouped by pairs and each pair can get configured to be either two single ended or a differential pair signal.	36
	Configuration pins or west GPIO bank WA alternatively The configuration interface consists of 18 pins. After configuration procedure, these balls can be configured to be normal GPIO signals. They are grouped by pairs and each pair can get configured to be either two single ended or a differential pair signal.	18
	West GPIO banks WB and WC Each bank has 18 GPIO signals and both have separate power supply balls and can be driven individually. GPIO signals are grouped by pairs and each pair can get configured to be either two single ended or a differential pair signal.	36
	SerDes signal pins The SerDes interface consits of differential transmit and receive signals and a termination input pin. Furthermore, dedicated power supply exists to the SerDes interface and the SerDes phase-locked loop (PLL) function.	5 / 10
	Power supply pins for FPGA core and GPIO The CCGM1A1 / CCGM1A2 core gets supplied from a single power source. The GPIO banks have own power supply pins each. Additionally, some further power supply pins are available for SerDes and other functions.	78
	Ground pins CCGM1A1 / CCGM1A2 has a single ground level for all supply voltages. All ground pins must be connected.	74 / 70
	Any other pins Clock input (single ended or differential), reset input, power-on reset adjustment input (CCGM1A1 only) and one not connected pin (must be left open) are available in this pin group.	5 / 4

5.2 CCGM1A1 Pin List

Table 5.3: *CCGM1A1 Pin list sorted by ball name*

Ball	Signal name	Signal group	Reset category	Description
A1	GND	Ground	0	Ground
A2	VDD_WC	Power	0	3rd GPIO west bank power supply
A3	IO_NA_A0	GPIO	1	1st GPIO north bank signal A0
A4	IO_NA_A1	GPIO	1	1st GPIO north bank signal A1
A5	VDD_NA	Power	0	1st GPIO north bank power supply
A6	IO_NA_A4	GPIO	1	1st GPIO north bank signal A4
A7	GND	Ground	0	Ground
A8	IO_NA_A7	GPIO	1	1st GPIO north bank signal A7
A9	IO_NB_B0	GPIO	1	2nd GPIO north bank signal B0
A10	GND	Ground	0	Ground
A11	IO_NB_B2	GPIO	1	2nd GPIO north bank signal B2
A12	IO_NB_B4	GPIO	1	2nd GPIO north bank signal B4
A13	GND	Ground	0	Ground
A14	IO_NB_B7	GPIO	1	2nd GPIO north bank signal B7
A15	IO_EB_B8	GPIO	1	2nd GPIO east bank signal B8
A16	VDD_EB	Power	0	2nd GPIO east bank power supply
A17	IO_EB_B5	GPIO	1	2nd GPIO east bank signal B5
A18	GND	Ground	0	Ground
B1	IO_WC_A8	GPIO	1	3rd GPIO west bank signal A8
B2	IO_WC_B8	GPIO	1	3rd GPIO west bank signal B8
B3	IO_NA_B0	GPIO	1	1st GPIO north bank signal B0
B4	IO_NA_B1	GPIO	1	1st GPIO north bank signal B1
B5	IO_NA_A2	GPIO	1	1st GPIO north bank signal A2
B6	IO_NA_B4	GPIO	1	1st GPIO north bank signal B4
B7	VDD_NA	Power	0	1st GPIO north bank power supply
B8	IO_NA_B7	GPIO	1	1st GPIO north bank signal B7
B9	IO_NB_A0	GPIO	1	2nd GPIO north bank signal A0
B10	VDD_NB	Power	0	2nd GPIO north bank power supply
B11	IO_NB_A2	GPIO	1	2nd GPIO north bank signal A2
B12	IO_NB_A4	GPIO	1	2nd GPIO north bank signal A4
B13	VDD_NB	Power	0	2nd GPIO north bank power supply
B14	IO_NB_A7	GPIO	1	2nd GPIO north bank signal A7
B15	IO_EB_A8	GPIO	1	2nd GPIO east bank signal A8

(continued on next page)

Table 5.3: *CCGM1A1 Pin list sorted by ball name*

(continued from previous page)

Ball	Signal name	Signal group	Reset category	Description
B16	GND	Ground	0	Ground
B17	IO_EB_A5	GPIO	1	2nd GPIO east bank signal A5
B18	VDD_EB	Power	0	2nd GPIO east bank power supply
C1	GND	Ground	0	Ground
C2	VDD_WC	Power	0	3rd GPIO west bank power supply
C3	IO_WC_A7	GPIO	1	3rd GPIO west bank signal A7
C4	IO_WC_B7	GPIO	1	3rd GPIO west bank signal B7
C5	IO_NA_B2	GPIO	1	1st GPIO north bank signal B2
C6	IO_NA_A3	GPIO	1	1st GPIO north bank signal A3
C7	IO_NA_A5	GPIO	1	1st GPIO north bank signal A5
C8	IO_NA_A6	GPIO	1	1st GPIO north bank signal A6
C9	IO_NA_A8	GPIO	1	1st GPIO north bank signal A8
C10	IO_NB_B1	GPIO	1	2nd GPIO north bank signal B1
C11	IO_NB_B3	GPIO	1	2nd GPIO north bank signal B3
C12	IO_NB_B5	GPIO	1	2nd GPIO north bank signal B5
C13	IO_NB_B6	GPIO	1	2nd GPIO north bank signal B6
C14	IO_NB_B8	GPIO	1	2nd GPIO north bank signal B8
C15	IO_EB_B7	GPIO	1	2nd GPIO east bank signal B7
C16	IO_EB_B6	GPIO	1	2nd GPIO east bank signal B6
C17	IO_EB_B4	GPIO	1	2nd GPIO east bank signal B4
C18	IO_EB_A4	GPIO	1	2nd GPIO east bank signal A4
D1	IO_WC_A5	GPIO	1	3rd GPIO west bank signal A5
D2	IO_WC_B5	GPIO	1	3rd GPIO west bank signal B5
D3	IO_WC_A6	GPIO	1	3rd GPIO west bank signal A6
D4	IO_WC_B6	GPIO	1	3rd GPIO west bank signal B6
D5	VDD_WC	Power	0	3rd GPIO west bank power supply
D6	IO_NA_B3	GPIO	1	1st GPIO north bank signal B3
D7	IO_NA_B5	GPIO	1	1st GPIO north bank signal B5
D8	IO_NA_B6	GPIO	1	1st GPIO north bank signal B6
D9	IO_NA_B8	GPIO	1	1st GPIO north bank signal B8
D10	IO_NB_A1	GPIO	1	2nd GPIO north bank signal A1
D11	IO_NB_A3	GPIO	1	2nd GPIO north bank signal A3
D12	IO_NB_A5	GPIO	1	2nd GPIO north bank signal A5
D13	IO_NB_A6	GPIO	1	2nd GPIO north bank signal A6
D14	IO_NB_A8	GPIO	1	2nd GPIO north bank signal A8

(continued on next page)

Table 5.3: *CCGM1A1 Pin list sorted by ball name*

(continued from previous page)

Ball	Signal name	Signal group	Reset category	Description
D15	IO_EB_A7	GPIO	1	2nd GPIO east bank signal A7
D16	IO_EB_A6	GPIO	1	2nd GPIO east bank signal A6
D17	IO_EB_B2	GPIO	1	2nd GPIO east bank signal B2
D18	IO_EB_A2	GPIO	1	2nd GPIO east bank signal A2
E1	IO_WC_A3	GPIO	1	3rd GPIO west bank signal A3
E2	IO_WC_B3	GPIO	1	3rd GPIO west bank signal B3
E3	IO_WC_A4	GPIO	1	3rd GPIO west bank signal A4
E4	IO_WC_B4	GPIO	1	3rd GPIO west bank signal B4
E5	GND	Ground	0	Ground
E6	VDD_NA	Power	0	1st GPIO north bank power supply
E7	GND	Ground	0	Ground
E8	VDD_NA	Power	0	1st GPIO north bank power supply
E9	GND	Ground	0	Ground
E10	VDD_NB	Power	0	2nd GPIO north bank power supply
E11	GND	Ground	0	Ground
E12	VDD_NB	Power	0	2nd GPIO north bank power supply
E13	GND	Ground	0	Ground
E14	VDD_EB	Power	0	2nd GPIO east bank power supply
E15	IO_EB_B3	GPIO	1	2nd GPIO east bank signal B3
E16	IO_EB_A3	GPIO	1	2nd GPIO east bank signal A3
E17	VDD_EB	Power	0	2nd GPIO east bank power supply
E18	GND	Ground	0	Ground
F1	GND	Ground	0	Ground
F2	VDD_WC	Power	0	3rd GPIO west bank power supply
F3	IO_WC_A2	GPIO	1	3rd GPIO west bank signal A2
F4	IO_WC_B2	GPIO	1	3rd GPIO west bank signal B2
F5	VDD_WC	Power	0	3rd GPIO west bank power supply
F6	GND	Ground	0	Ground
F7	VDD_NA	Power	0	1st GPIO north bank power supply
F8	GND	Ground	0	Ground
F9	VDD	Power	0	Core power supply
F10	GND	Ground	0	Ground
F11	VDD_NB	Power	0	2nd GPIO north bank power supply
F12	GND	Ground	0	Ground
F13	VDD_EB	Power	0	2nd GPIO east bank power supply

(continued on next page)

Table 5.3: *CCGM1A1 Pin list sorted by ball name*

(continued from previous page)

Ball	Signal name	Signal group	Reset category	Description
F14	GND	Ground	0	Ground
F15	IO_EB_B1	GPIO	1	2nd GPIO east bank signal B1
F16	IO_EB_A1	GPIO	1	2nd GPIO east bank signal A1
F17	IO_EB_B0	GPIO	1	2nd GPIO east bank signal B0
F18	IO_EB_A0	GPIO	1	2nd GPIO east bank signal A0
G1	IO_WC_A0	GPIO	1	3rd GPIO west bank signal A0
G2	IO_WC_B0	GPIO	1	3rd GPIO west bank signal B0
G3	IO_WC_A1	GPIO	1	3rd GPIO west bank signal A1
G4	IO_WC_B1	GPIO	1	3rd GPIO west bank signal B1
G5	GND	Ground	0	Ground
G6	VDD	Power	0	Core power supply
G7	GND	Ground	0	Ground
G8	VDD	Power	0	Core power supply
G9	GND	Ground	0	Ground
G10	VDD	Power	0	Core power supply
G11	GND	Ground	0	Ground
G12	VDD	Power	0	Core power supply
G13	GND	Ground	0	Ground
G14	VDD_EA	Power	0	1st GPIO east bank power supply
G15	IO_EA_B8	GPIO	1	1st GPIO east bank signal B8
G16	IO_EA_A8	GPIO	1	1st GPIO east bank signal A8
G17	IO_EA_B7	GPIO	1	1st GPIO east bank signal B7
G18	IO_EA_A7	GPIO	1	1st GPIO east bank signal A7
H1	IO_WB_A7	GPIO	1	2nd GPIO west bank signal A7
H2	IO_WB_B7	GPIO	1	2nd GPIO west bank signal B7
H3	IO_WB_A8	GPIO	1	2nd GPIO west bank signal A8
H4	IO_WB_B8	GPIO	1	2nd GPIO west bank signal B8
H5	VDD_WB	Power	0	2nd GPIO west bank power supply
H6	GND	Ground	0	Ground
H7	VDD	Power	0	Core power supply
H8	GND	Ground	0	Ground
H9	VDD	Power	0	Core power supply
H10	GND	Ground	0	Ground
H11	VDD	Power	0	Core power supply
H12	GND	Ground	0	Ground

(continued on next page)

Table 5.3: *CCGM1A1 Pin list sorted by ball name*

(continued from previous page)

Ball	Signal name	Signal group	Reset category	Description
H13	VDD	Power	0	Core power supply
H14	GND	Ground	0	Ground
H15	IO_EA_B6	GPIO	1	1st GPIO east bank signal B6
H16	IO_EA_A6	GPIO	1	1st GPIO east bank signal A6
H17	VDD_EA	Power	0	1st GPIO east bank power supply
H18	GND	Ground	0	Ground
J1	GND	Ground	0	Ground
J2	VDD_WB	Power	0	2nd GPIO west bank power supply
J3	IO_WB_A6	GPIO	1	2nd GPIO west bank signal A6
J4	IO_WB_B6	GPIO	1	2nd GPIO west bank signal B6
J5	GND	Ground	0	Ground
J6	VDD	Power	0	Core power supply
J7	GND	Ground	0	Ground
J8	VDD	Power	0	Core power supply
J9	GND	Ground	0	Ground
J10	VDD	Power	0	Core power supply
J11	GND	Ground	0	Ground
J12	VDD	Power	0	Core power supply
J13	GND	Ground	0	Ground
J14	VDD_EA	Power	0	1st GPIO east bank power supply
J15	IO_EA_B5	GPIO	1	1st GPIO east bank signal B5
J16	IO_EA_A5	GPIO	1	1st GPIO east bank signal A5
J17	IO_EA_B4	GPIO	1	1st GPIO east bank signal B4
J18	IO_EA_A4	GPIO	1	1st GPIO east bank signal A4
K1	IO_WB_A5	GPIO	1	2nd GPIO west bank signal A5
K2	IO_WB_B5	GPIO	1	2nd GPIO west bank signal B5
K3	IO_WB_A4	GPIO	1	2nd GPIO west bank signal A4
K4	IO_WB_B4	GPIO	1	2nd GPIO west bank signal B4
K5	VDD_WB	Power	0	2nd GPIO west bank power supply
K6	GND	Ground	0	Ground
K7	VDD	Power	0	Core power supply
K8	GND	Ground	0	Ground
K9	VDD	Power	0	Core power supply
K10	GND	Ground	0	Ground
K11	VDD	Power	0	Core power supply

(continued on next page)

Table 5.3: *CCGM1A1 Pin list sorted by ball name*

(continued from previous page)

Ball	Signal name	Signal group	Reset category	Description
K12	GND	Ground	0	Ground
K13	VDD	Power	0	Core power supply
K14	GND	Ground	0	Ground
K15	IO_EA_B3	GPIO	1	1st GPIO east bank signal B3
K16	IO_EA_A3	GPIO	1	1st GPIO east bank signal A3
K17	IO_EA_B2	GPIO	1	1st GPIO east bank signal B2
K18	IO_EA_A2	GPIO	1	1st GPIO east bank signal A2
L1	IO_WB_A3	GPIO	1	2nd GPIO west bank signal A3
L2	IO_WB_B3	GPIO	1	2nd GPIO west bank signal B3
L3	IO_WB_A2	GPIO	1	2nd GPIO west bank signal A2
L4	IO_WB_B2	GPIO	1	2nd GPIO west bank signal B2
L5	GND	Ground	0	Ground
L6	VDD	Power	0	Core power supply
L7	GND	Ground	0	Ground
L8	VDD	Power	0	Core power supply
L9	GND	Ground	0	Ground
L10	VDD	Power	0	Core power supply
L11	GND	Ground	0	Ground
L12	VDD	Power	0	Core power supply
L13	GND	Ground	0	Ground
L14	VDD_EA	Power	0	1st GPIO east bank power supply
L15	IO_EA_B1	GPIO	1	1st GPIO east bank signal B1
L16	IO_EA_A1	GPIO	1	1st GPIO east bank signal A1
L17	VDD_EA	Power	0	1st GPIO east bank power supply
L18	GND	Ground	0	Ground
M1	GND	Ground	0	Ground
M2	VDD_WB	Power	0	2nd GPIO west bank power supply
M3	IO_WB_A1	GPIO	1	2nd GPIO west bank signal A1
M4	IO_WB_B1	GPIO	1	2nd GPIO west bank signal B1
M5	VDD_WB	Power	0	2nd GPIO west bank power supply
M6	GND	Ground	0	Ground
M7	VDD	Power	0	Core power supply
M8	GND	Ground	0	Ground
M9	VDD	Power	0	Core power supply
M10	GND	Ground	0	Ground

(continued on next page)

Table 5.3: *CCGM1A1 Pin list sorted by ball name*

(continued from previous page)

Ball	Signal name	Signal group	Reset category	Description
M11	VDD	Power	0	Core power supply
M12	GND	Ground	0	Ground
M13	VDD	Power	0	Core power supply
M14	IO_EA_B0	GPIO	1	1st GPIO east bank signal B0
M15	IO_EA_A0	GPIO	1	1st GPIO east bank signal A0
M16	GND	Ground	0	Ground
M17	IO_SB_A3	GPIO	1	2nd GPIO south bank signal A3
M18	IO_SB_B3	GPIO	1	2nd GPIO south bank signal B3
N1	IO_WB_A0	GPIO	1	2nd GPIO west bank signal A0
N2	IO_WB_B0	GPIO	1	2nd GPIO west bank signal B0
N3	SPI_CS_N	GPIO	3	1st function: Configuration SPI chip select
N3	IO_WA_A8	GPIO	1	2nd function: 1st GPIO west bank signal A8
N4	SPI_CLK	GPIO	3	1st function: Configuration SPI clock
N4	IO_WA_B8	GPIO	1	2nd function: 1st GPIO west bank signal B8
N5	VDD_WA	Power	0	1st GPIO west bank power supply
N6	VDD	Power	0	Core power supply
N7	GND	Ground	0	Ground
N8	VDD	Power	0	Core power supply
N9	GND	Ground	0	Ground
N10	VDD	Power	0	Core power supply
N11	VDD_SB	Power	0	2nd GPIO south bank power supply
N12	GND	Ground	0	Ground
N13	VDD_SB	Power	0	2nd GPIO south bank power supply
N14	IO_SB_A8	GPIO	1	1st function: 2nd GPIO south bank signal A8
N14	CLK0	GPIO	1	2nd function: 1st clock input
N15	IO_SB_B8	GPIO	1	2nd GPIO south bank signal B8
N16	N.C.	Other	0	Not connected. This pin must be left open.
N17	GND	Ground	0	Ground
N18	VDD_SB	Power	0	2nd GPIO south bank power supply
P1	SPI_D1	GPIO	4	1st function: Configuration SPI data bit 1
P1	IO_WA_A7	GPIO	1	2nd function: 1st GPIO west bank signal A7
P2	SPI_D0	GPIO	3	1st function: Configuration SPI data bit 0
P2	IO_WA_B7	GPIO	1	2nd function: 1st GPIO west bank signal B7

(continued on next page)

Table 5.3: *CCGM1A1 Pin list sorted by ball name*

(continued from previous page)

Ball	Signal name	Signal group	Reset category	Description
P3	VDD_WA	Power	0	1st GPIO west bank power supply
P4	GND	Ground	0	Ground
P5	VDD_WA	Power	0	1st GPIO west bank power supply
P6	VDD_SA	Power	0	1st GPIO south bank power supply
P7	GND	Ground	0	Ground
P8	VDD_SA	Power	0	1st GPIO south bank power supply
P9	GND	Ground	0	Ground
P10	VDD_SA	Power	0	1st GPIO south bank power supply
P11	IO_SB_A4	GPIO	1	2nd GPIO south bank signal A4
P12	IO_SB_A7	GPIO	1	1st function: 2nd GPIO south bank signal A7
P12	CLK1	GPIO	1	2nd function: 2nd clock input
P13	IO_SB_B7	GPIO	1	2nd GPIO south bank signal B7
P14	IO_SB_A6	GPIO	1	1st function: 2nd GPIO south bank signal A6
P14	CLK2	GPIO	1	2nd function: 3rd clock input
P15	IO_SB_B6	GPIO	1	2nd GPIO south bank signal B6
P16	VDD_PLL	Power	0	PLL power supply
P17	IO_SB_A2	GPIO	1	2nd GPIO south bank signal A2
P18	IO_SB_B2	GPIO	1	2nd GPIO south bank signal B2
R1	SPI_D3	GPIO	4	1st function: Configuration SPI data bit 3
R1	IO_WA_A6	GPIO	1	2nd function: 1st GPIO west bank signal A6
R2	SPI_D2	GPIO	4	1st function: Configuration SPI data bit 2
R2	IO_WA_B6	GPIO	1	2nd function: 1st GPIO west bank signal B6
R3	JTAG_TCK	GPIO	4	1st function: Configuration JTAG clock
R3	IO_WA_A5	GPIO	1	2nd function: 1st GPIO west bank signal A5
R4	SPI_FWD	GPIO	2	1st function: Configuration SPI data forward
R4	IO_WA_B5	GPIO	1	2nd function: 1st GPIO west bank signal B5
R5	CFG_MD0	GPIO	4	1st function: Configuration mode bit 0
R5	IO_WA_A0	GPIO	1	2nd function: 1st GPIO west bank signal A0
R6	IO_SA_A1	GPIO	1	1st GPIO south bank signal A1
R7	IO_SA_A2	GPIO	1	1st GPIO south bank signal A2
R8	IO_SA_A4	GPIO	1	1st GPIO south bank signal A4
R9	IO_SA_A6	GPIO	1	1st GPIO south bank signal A6
R10	IO_SA_A7	GPIO	1	1st GPIO south bank signal A7

(continued on next page)

Table 5.3: *CCGM1A1 Pin list sorted by ball name*

(continued from previous page)

Ball	Signal name	Signal group	Reset category	Description
R11	IO_SB_B4	GPIO	1	2nd GPIO south bank signal B4
R12	GND	Ground	0	Ground
R13	IO_SB_A5	GPIO	1	1st function: 2nd GPIO south bank signal A5
R13	CLK3	GPIO	1	2nd function: 4th clock input
R14	IO_SB_B5	GPIO	1	2nd GPIO south bank signal B5
R15	VDD_SB	Power	0	2nd GPIO south bank power supply
R16	GND	Ground	0	Ground
R17	IO_SB_A1	GPIO	1	2nd GPIO south bank signal A1
R18	IO_SB_B1	GPIO	1	2nd GPIO south bank signal B1
T1	VDD_WA	Power	0	1st GPIO west bank power supply
T2	JTAG_TDI	GPIO	4	1st function: JTAG data input
T2	IO_WA_A4	GPIO	1	2nd function: 1st GPIO west bank signal A4
T3	JTAG_TMS	GPIO	4	1st function: JTAG test mode select
T3	IO_WA_B4	GPIO	1	2nd function: 1st GPIO west bank signal B4
T4	GND	Ground	0	Ground
T5	CFG_MD1	GPIO	4	1st function: Configuration mode bit 1
T5	IO_WA_B0	GPIO	1	2nd function: 1st GPIO west bank signal B0
T6	IO_SA_B1	GPIO	1	1st GPIO south bank signal B1
T7	IO_SA_B2	GPIO	1	1st GPIO south bank signal B2
T8	IO_SA_B4	GPIO	1	1st GPIO south bank signal B4
T9	IO_SA_B6	GPIO	1	1st GPIO south bank signal B6
T10	IO_SA_B7	GPIO	1	1st GPIO south bank signal B7
T11	GND	Ground	0	Ground
T12	SER_CLK	Other	4	SerDes clock, positive LVDS signal (or single ended)
T13	SER_CLK_N	Other	1	SerDes clock, negative LVDS signal
T14	VDD_CLK	Power	0	Clock signal power supply
T15	RST_N	Other	4	Reset
T16	VDD_SER_PLL	Power	0	SerDes PLL power supply (1.0 V to 1.1 V ±50 mV)
T17	GND	Ground	0	Ground
T18	VDD_SB	Power	0	2nd GPIO south bank power supply
U1	POR_EN	GPIO	2	1st function: Enable power-on reset

(continued on next page)

Table 5.3: *CCGM1A1 Pin list sorted by ball name*

(continued from previous page)

Ball	Signal name	Signal group	Reset category	Description
U1	IO_WA_A3	GPIO	1	2nd function: 1st GPIO west bank signal A3
U2	JTAG_TDO	GPIO	2	1st function: JTAG data output
U2	IO_WA_B3	GPIO	1	2nd function: 1st GPIO west bank signal B3
U3	VDD_WA	Power	0	1st GPIO west bank power supply
U4	CFG_MD2	GPIO	4	1st function: Configuration mode bit 2
U4	IO_WA_A1	GPIO	1	2nd function: 1st GPIO west bank signal A1
U5	IO_SA_A0	GPIO	1	1st GPIO south bank signal A0
U6	VDD_SA	Power	0	1st GPIO south bank power supply
U7	IO_SA_A3	GPIO	1	1st GPIO south bank signal A3
U8	IO_SA_A5	GPIO	1	1st GPIO south bank signal A5
U9	VDD_SA	Power	0	1st GPIO south bank power supply
U10	IO_SA_A8	GPIO	1	1st GPIO south bank signal A8
U11	SER_RX_P	SerDes	1	Receive data line, positive LVDS signal
U12	VDD_SER	Power	0	SerDes core power supply (1.0 V to 1.1 V ±50 mV)
U13	SER_TX_P	SerDes	1	Transmit data line, positive LVDS signal
U14	GND	Ground	0	Ground
U15	GND	Ground	0	Ground
U16	GND	Ground	0	Ground
U17	IO_SB_A0	GPIO	1	2nd GPIO south bank signal A0
U18	IO_SB_B0	GPIO	1	2nd GPIO south bank signal B0
V1	GND	Ground	0	Ground
V2	CFG_FAILED_N	GPIO	2	1st function: Configuration failed signal
V2	IO_WA_A2	GPIO	1	2nd function: 1st GPIO west bank signal A2
V3	CFG_DONE	GPIO	2	1st function: Configuration done signal
V3	IO_WA_B2	GPIO	1	2nd function: 1st GPIO west bank signal B2
V4	CFG_MD3	GPIO	4	1st function: Configuration mode bit 3
V4	IO_WA_B1	GPIO	1	2nd function: 1st GPIO west bank signal B1
V5	IO_SA_B0	GPIO	1	1st GPIO south bank signal B0
V6	GND	Ground	0	Ground
V7	IO_SA_B3	GPIO	1	1st GPIO south bank signal B3
V8	IO_SA_B5	GPIO	1	1st GPIO south bank signal B5
V9	GND	Ground	0	Ground

(continued on next page)

Table 5.3: *CCGM1A1 Pin list sorted by ball name*

(continued from previous page)

Ball	Signal name	Signal group	Reset category	Description
V10	IO_SA_B8	GPIO	1	1st GPIO south bank signal B8
V11	SER_RX_N	SerDes	1	Receive data line, negative LVDS signal
V12	SER_RTERM	SerDes	0	Line termination
V13	SER_TX_N	SerDes	1	Transmit data line, negative LVDS signal
V14	GND	Ground	0	Ground
V15	POR_ADJ	Other	0	Power-on-Reset level adjustment
V16	GND	Ground	0	Ground
V17	VDD_SER	Power	0	SerDes core power supply (1.0 V to 1.1 V ±50 mV)
V18	GND	Ground	0	Ground

Reset categories:

- **0:** No reset (supply or analog input pin)
- **1:** Pin disabled, high-z
- **2:** Pin characteristic already during reset state
- **3:** Pin characteristic depends on the configuration mode (SPI master: I/O, else: input)
- **4:** Input pin

5.3 CCGM1A2 Pin List

Table 5.4: *CCGM1A2 Pin list sorted by ball name*

Ball	Signal name	Signal group	Reset category	Description
A1	GND	Ground	0	Ground
A2	VDD_WC	Power	0	3rd GPIO west bank power supply
A3	IO_NA_A0	GPIO	1	1st GPIO north bank signal A0
A4	IO_NA_A1	GPIO	1	1st GPIO north bank signal A1
A5	VDD_NA	Power	0	1st GPIO north bank power supply
A6	IO_NA_A4	GPIO	1	1st GPIO north bank signal A4
A7	GND	Ground	0	Ground
A8	IO_NA_A7	GPIO	1	1st GPIO north bank signal A7
A9	IO_NB_B0	GPIO	1	2nd GPIO north bank signal B0
A10	GND	Ground	0	Ground
A11	IO_NB_B2	GPIO	1	2nd GPIO north bank signal B2
A12	IO_NB_B4	GPIO	1	2nd GPIO north bank signal B4
A13	GND	Ground	0	Ground
A14	IO_NB_B7	GPIO	1	2nd GPIO north bank signal B7
A15	IO_EB_B8	GPIO	1	2nd GPIO east bank signal B8
A16	VDD_EB	Power	0	2nd GPIO east bank power supply
A17	IO_EB_B5	GPIO	1	2nd GPIO east bank signal B5
A18	GND	Ground	0	Ground
B1	IO_WC_A8	GPIO	1	3rd GPIO west bank signal A8
B2	IO_WC_B8	GPIO	1	3rd GPIO west bank signal B8
B3	IO_NA_B0	GPIO	1	1st GPIO north bank signal B0
B4	IO_NA_B1	GPIO	1	1st GPIO north bank signal B1
B5	IO_NA_A2	GPIO	1	1st GPIO north bank signal A2
B6	IO_NA_B4	GPIO	1	1st GPIO north bank signal B4
B7	VDD_NA	Power	0	1st GPIO north bank power supply
B8	IO_NA_B7	GPIO	1	1st GPIO north bank signal B7
B9	IO_NB_A0	GPIO	1	2nd GPIO north bank signal A0
B10	VDD_NB	Power	0	2nd GPIO north bank power supply
B11	IO_NB_A2	GPIO	1	2nd GPIO north bank signal A2
B12	IO_NB_A4	GPIO	1	2nd GPIO north bank signal A4
B13	VDD_NB	Power	0	2nd GPIO north bank power supply
B14	IO_NB_A7	GPIO	1	2nd GPIO north bank signal A7
B15	IO_EB_A8	GPIO	1	2nd GPIO east bank signal A8

(continued on next page)

Table 5.4: *CCGM1A2 Pin list sorted by ball name*

(continued from previous page)

Ball	Signal name	Signal group	Reset category	Description
B16	GND	Ground	0	Ground
B17	IO_EB_A5	GPIO	1	2nd GPIO east bank signal A5
B18	VDD_EB	Power	0	2nd GPIO east bank power supply
C1	GND	Ground	0	Ground
C2	VDD_WC	Power	0	3rd GPIO west bank power supply
C3	IO_WC_A7	GPIO	1	3rd GPIO west bank signal A7
C4	IO_WC_B7	GPIO	1	3rd GPIO west bank signal B7
C5	IO_NA_B2	GPIO	1	1st GPIO north bank signal B2
C6	IO_NA_A3	GPIO	1	1st GPIO north bank signal A3
C7	IO_NA_A5	GPIO	1	1st GPIO north bank signal A5
C8	IO_NA_A6	GPIO	1	1st GPIO north bank signal A6
C9	IO_NA_A8	GPIO	1	1st GPIO north bank signal A8
C10	IO_NB_B1	GPIO	1	2nd GPIO north bank signal B1
C11	IO_NB_B3	GPIO	1	2nd GPIO north bank signal B3
C12	IO_NB_B5	GPIO	1	2nd GPIO north bank signal B5
C13	IO_NB_B6	GPIO	1	2nd GPIO north bank signal B6
C14	IO_NB_B8	GPIO	1	2nd GPIO north bank signal B8
C15	IO_EB_B7	GPIO	1	2nd GPIO east bank signal B7
C16	IO_EB_B6	GPIO	1	2nd GPIO east bank signal B6
C17	IO_EB_B4	GPIO	1	2nd GPIO east bank signal B4
C18	IO_EB_A4	GPIO	1	2nd GPIO east bank signal A4
D1	IO_WC_A5	GPIO	1	3rd GPIO west bank signal A5
D2	IO_WC_B5	GPIO	1	3rd GPIO west bank signal B5
D3	IO_WC_A6	GPIO	1	3rd GPIO west bank signal A6
D4	IO_WC_B6	GPIO	1	3rd GPIO west bank signal B6
D5	VDD_WC	Power	0	3rd GPIO west bank power supply
D6	IO_NA_B3	GPIO	1	1st GPIO north bank signal B3
D7	IO_NA_B5	GPIO	1	1st GPIO north bank signal B5
D8	IO_NA_B6	GPIO	1	1st GPIO north bank signal B6
D9	IO_NA_B8	GPIO	1	1st GPIO north bank signal B8
D10	IO_NB_A1	GPIO	1	2nd GPIO north bank signal A1
D11	IO_NB_A3	GPIO	1	2nd GPIO north bank signal A3
D12	IO_NB_A5	GPIO	1	2nd GPIO north bank signal A5
D13	IO_NB_A6	GPIO	1	2nd GPIO north bank signal A6
D14	IO_NB_A8	GPIO	1	2nd GPIO north bank signal A8

(continued on next page)

Table 5.4: *CCGM1A2 Pin list sorted by ball name*

(continued from previous page)

Ball	Signal name	Signal group	Reset category	Description
D15	IO_EB_A7	GPIO	1	2nd GPIO east bank signal A7
D16	IO_EB_A6	GPIO	1	2nd GPIO east bank signal A6
D17	IO_EB_B2	GPIO	1	2nd GPIO east bank signal B2
D18	IO_EB_A2	GPIO	1	2nd GPIO east bank signal A2
E1	IO_WC_A3	GPIO	1	3rd GPIO west bank signal A3
E2	IO_WC_B3	GPIO	1	3rd GPIO west bank signal B3
E3	IO_WC_A4	GPIO	1	3rd GPIO west bank signal A4
E4	IO_WC_B4	GPIO	1	3rd GPIO west bank signal B4
E5	GND	Ground	0	Ground
E6	VDD_NA	Power	0	1st GPIO north bank power supply
E7	GND	Ground	0	Ground
E8	VDD_NA	Power	0	1st GPIO north bank power supply
E9	GND	Ground	0	Ground
E10	VDD_NB	Power	0	2nd GPIO north bank power supply
E11	GND	Ground	0	Ground
E12	VDD_NB	Power	0	2nd GPIO north bank power supply
E13	GND	Ground	0	Ground
E14	VDD_EB	Power	0	2nd GPIO east bank power supply
E15	IO_EB_B3	GPIO	1	2nd GPIO east bank signal B3
E16	IO_EB_A3	GPIO	1	2nd GPIO east bank signal A3
E17	VDD_EB	Power	0	2nd GPIO east bank power supply
E18	GND	Ground	0	Ground
F1	GND	Ground	0	Ground
F2	VDD_WC	Power	0	3rd GPIO west bank power supply
F3	IO_WC_A2	GPIO	1	3rd GPIO west bank signal A2
F4	IO_WC_B2	GPIO	1	3rd GPIO west bank signal B2
F5	VDD_WC	Power	0	3rd GPIO west bank power supply
F6	GND	Ground	0	Ground
F7	VDD_NA	Power	0	1st GPIO north bank power supply
F8	GND	Ground	0	Ground
F9	VDD	Power	0	Core power supply
F10	GND	Ground	0	Ground
F11	VDD_NB	Power	0	2nd GPIO north bank power supply
F12	GND	Ground	0	Ground
F13	VDD_EB	Power	0	2nd GPIO east bank power supply

(continued on next page)

Table 5.4: *CCGM1A2 Pin list sorted by ball name*

(continued from previous page)

Ball	Signal name	Signal group	Reset category	Description
F14	GND	Ground	0	Ground
F15	IO_EB_B1	GPIO	1	2nd GPIO east bank signal B1
F16	IO_EB_A1	GPIO	1	2nd GPIO east bank signal A1
F17	IO_EB_B0	GPIO	1	2nd GPIO east bank signal B0
F18	IO_EB_A0	GPIO	1	2nd GPIO east bank signal A0
G1	IO_WC_A0	GPIO	1	3rd GPIO west bank signal A0
G2	IO_WC_B0	GPIO	1	3rd GPIO west bank signal B0
G3	IO_WC_A1	GPIO	1	3rd GPIO west bank signal A1
G4	IO_WC_B1	GPIO	1	3rd GPIO west bank signal B1
G5	GND	Ground	0	Ground
G6	VDD	Power	0	Core power supply
G7	GND	Ground	0	Ground
G8	VDD	Power	0	Core power supply
G9	GND	Ground	0	Ground
G10	VDD	Power	0	Core power supply
G11	GND	Ground	0	Ground
G12	VDD	Power	0	Core power supply
G13	GND	Ground	0	Ground
G14	VDD_EA	Power	0	1st GPIO east bank power supply
G15	IO_EA_B8	GPIO	1	1st GPIO east bank signal B8
G16	IO_EA_A8	GPIO	1	1st GPIO east bank signal A8
G17	IO_EA_B7	GPIO	1	1st GPIO east bank signal B7
G18	IO_EA_A7	GPIO	1	1st GPIO east bank signal A7
H1	IO_WB_A7	GPIO	1	2nd GPIO west bank signal A7
H2	IO_WB_B7	GPIO	1	2nd GPIO west bank signal B7
H3	IO_WB_A8	GPIO	1	2nd GPIO west bank signal A8
H4	IO_WB_B8	GPIO	1	2nd GPIO west bank signal B8
H5	VDD_WB	Power	0	2nd GPIO west bank power supply
H6	GND	Ground	0	Ground
H7	VDD	Power	0	Core power supply
H8	GND	Ground	0	Ground
H9	VDD	Power	0	Core power supply
H10	GND	Ground	0	Ground
H11	VDD	Power	0	Core power supply
H12	GND	Ground	0	Ground

(continued on next page)

Table 5.4: *CCGM1A2 Pin list sorted by ball name*

(continued from previous page)

Ball	Signal name	Signal group	Reset category	Description
H13	VDD	Power	0	Core power supply
H14	GND	Ground	0	Ground
H15	IO_EA_B6	GPIO	1	1st GPIO east bank signal B6
H16	IO_EA_A6	GPIO	1	1st GPIO east bank signal A6
H17	VDD_EA	Power	0	1st GPIO east bank power supply
H18	GND	Ground	0	Ground
J1	GND	Ground	0	Ground
J2	VDD_WB	Power	0	2nd GPIO west bank power supply
J3	IO_WB_A6	GPIO	1	2nd GPIO west bank signal A6
J4	IO_WB_B6	GPIO	1	2nd GPIO west bank signal B6
J5	GND	Ground	0	Ground
J6	VDD	Power	0	Core power supply
J7	GND	Ground	0	Ground
J8	VDD	Power	0	Core power supply
J9	GND	Ground	0	Ground
J10	VDD	Power	0	Core power supply
J11	GND	Ground	0	Ground
J12	VDD	Power	0	Core power supply
J13	GND	Ground	0	Ground
J14	VDD_EA	Power	0	1st GPIO east bank power supply
J15	IO_EA_B5	GPIO	1	1st GPIO east bank signal B5
J16	IO_EA_A5	GPIO	1	1st GPIO east bank signal A5
J17	IO_EA_B4	GPIO	1	1st GPIO east bank signal B4
J18	IO_EA_A4	GPIO	1	1st GPIO east bank signal A4
K1	IO_WB_A5	GPIO	1	2nd GPIO west bank signal A5
K2	IO_WB_B5	GPIO	1	2nd GPIO west bank signal B5
K3	IO_WB_A4	GPIO	1	2nd GPIO west bank signal A4
K4	IO_WB_B4	GPIO	1	2nd GPIO west bank signal B4
K5	VDD_WB	Power	0	2nd GPIO west bank power supply
K6	GND	Ground	0	Ground
K7	VDD	Power	0	Core power supply
K8	GND	Ground	0	Ground
K9	VDD	Power	0	Core power supply
K10	GND	Ground	0	Ground
K11	VDD	Power	0	Core power supply

(continued on next page)

Table 5.4: *CCGM1A2 Pin list sorted by ball name*

(continued from previous page)

Ball	Signal name	Signal group	Reset category	Description
K12	GND	Ground	0	Ground
K13	VDD	Power	0	Core power supply
K14	GND	Ground	0	Ground
K15	IO_EA_B3	GPIO	1	1st GPIO east bank signal B3
K16	IO_EA_A3	GPIO	1	1st GPIO east bank signal A3
K17	IO_EA_B2	GPIO	1	1st GPIO east bank signal B2
K18	IO_EA_A2	GPIO	1	1st GPIO east bank signal A2
L1	IO_WB_A3	GPIO	1	2nd GPIO west bank signal A3
L2	IO_WB_B3	GPIO	1	2nd GPIO west bank signal B3
L3	IO_WB_A2	GPIO	1	2nd GPIO west bank signal A2
L4	IO_WB_B2	GPIO	1	2nd GPIO west bank signal B2
L5	GND	Ground	0	Ground
L6	VDD	Power	0	Core power supply
L7	GND	Ground	0	Ground
L8	VDD	Power	0	Core power supply
L9	GND	Ground	0	Ground
L10	VDD	Power	0	Core power supply
L11	GND	Ground	0	Ground
L12	VDD	Power	0	Core power supply
L13	GND	Ground	0	Ground
L14	VDD_EA	Power	0	1st GPIO east bank power supply
L15	IO_EA_B1	GPIO	1	1st GPIO east bank signal B1
L16	IO_EA_A1	GPIO	1	1st GPIO east bank signal A1
L17	VDD_EA	Power	0	1st GPIO east bank power supply
L18	GND	Ground	0	Ground
M1	GND	Ground	0	Ground
M2	VDD_WB	Power	0	2nd GPIO west bank power supply
M3	IO_WB_A1	GPIO	1	2nd GPIO west bank signal A1
M4	IO_WB_B1	GPIO	1	2nd GPIO west bank signal B1
M5	VDD_WB	Power	0	2nd GPIO west bank power supply
M6	GND	Ground	0	Ground
M7	VDD	Power	0	Core power supply
M8	GND	Ground	0	Ground
M9	VDD	Power	0	Core power supply
M10	GND	Ground	0	Ground

(continued on next page)

Table 5.4: *CCGM1A2 Pin list sorted by ball name*

(continued from previous page)

Ball	Signal name	Signal group	Reset category	Description
M11	VDD	Power	0	Core power supply
M12	GND	Ground	0	Ground
M13	VDD	Power	0	Core power supply
M14	IO_EA_B0	GPIO	1	1st GPIO east bank signal B0
M15	IO_EA_A0	GPIO	1	1st GPIO east bank signal A0
M16	GND	Ground	0	Ground
M17	IO_SB_A3	GPIO	1	2nd GPIO south bank signal A3
M18	IO_SB_B3	GPIO	1	2nd GPIO south bank signal B3
N1	IO_WB_A0	GPIO	1	2nd GPIO west bank signal A0
N2	IO_WB_B0	GPIO	1	2nd GPIO west bank signal B0
N3	SPI_CS_N	GPIO	3	1st function: Configuration SPI chip select
N3	IO_WA_A8	GPIO	1	2nd function: 1st GPIO west bank signal A8
N4	SPI_CLK	GPIO	3	1st function: Configuration SPI clock
N4	IO_WA_B8	GPIO	1	2nd function: 1st GPIO west bank signal B8
N5	VDD_WA	Power	0	1st GPIO west bank power supply
N6	VDD	Power	0	Core power supply
N7	GND	Ground	0	Ground
N8	VDD	Power	0	Core power supply
N9	GND	Ground	0	Ground
N10	VDD	Power	0	Core power supply
N11	VDD_SB	Power	0	2nd GPIO south bank power supply
N12	GND	Ground	0	Ground
N13	VDD_SB	Power	0	2nd GPIO south bank power supply
N14	IO_SB_A8	GPIO	1	1st function: 2nd GPIO south bank signal A8
N14	CLK0	GPIO	1	2nd function: 1st clock input
N15	IO_SB_B8	GPIO	1	2nd GPIO south bank signal B8
N16	N.C.	Other	0	Not connected. This pin must be left open.
N17	GND	Ground	0	Ground
N18	VDD_SB	Power	0	2nd GPIO south bank power supply
P1	SPI_D1	GPIO	4	1st function: Configuration SPI data bit 1
P1	IO_WA_A7	GPIO	1	2nd function: 1st GPIO west bank signal A7
P2	SPI_D0	GPIO	3	1st function: Configuration SPI data bit 0
P2	IO_WA_B7	GPIO	1	2nd function: 1st GPIO west bank signal B7

(continued on next page)

Table 5.4: *CCGM1A2 Pin list sorted by ball name*

(continued from previous page)

Ball	Signal name	Signal group	Reset category	Description
P3	VDD_WA	Power	0	1st GPIO west bank power supply
P4	GND	Ground	0	Ground
P5	VDD_WA	Power	0	1st GPIO west bank power supply
P6	VDD_SA	Power	0	1st GPIO south bank power supply
P7	GND	Ground	0	Ground
P8	VDD_SA	Power	0	1st GPIO south bank power supply
P9	GND	Ground	0	Ground
P10	VDD_SA	Power	0	1st GPIO south bank power supply
P11	IO_SB_A4	GPIO	1	2nd GPIO south bank signal A4
P12	IO_SB_A7	GPIO	1	1st function: 2nd GPIO south bank signal A7
P12	CLK1	GPIO	1	2nd function: 2nd clock input
P13	IO_SB_B7	GPIO	1	2nd GPIO south bank signal B7
P14	IO_SB_A6	GPIO	1	1st function: 2nd GPIO south bank signal A6
P14	CLK2	GPIO	1	2nd function: 3rd clock input
P15	IO_SB_B6	GPIO	1	2nd GPIO south bank signal B6
P16	VDD_PLL	Power	0	PLL power supply
P17	IO_SB_A2	GPIO	1	2nd GPIO south bank signal A2
P18	IO_SB_B2	GPIO	1	2nd GPIO south bank signal B2
R1	SPI_D3	GPIO	4	1st function: Configuration SPI data bit 3
R1	IO_WA_A6	GPIO	1	2nd function: 1st GPIO west bank signal A6
R2	SPI_D2	GPIO	4	1st function: Configuration SPI data bit 2
R2	IO_WA_B6	GPIO	1	2nd function: 1st GPIO west bank signal B6
R3	JTAG_TCK	GPIO	4	1st function: Configuration JTAG clock
R3	IO_WA_A5	GPIO	1	2nd function: 1st GPIO west bank signal A5
R4	SPI_FWD	GPIO	2	1st function: Configuration SPI data forward
R4	IO_WA_B5	GPIO	1	2nd function: 1st GPIO west bank signal B5
R5	CFG_MD0	GPIO	4	1st function: Configuration mode bit 0
R5	IO_WA_A0	GPIO	1	2nd function: 1st GPIO west bank signal A0
R6	IO_SA_A1	GPIO	1	1st GPIO south bank signal A1
R7	IO_SA_A2	GPIO	1	1st GPIO south bank signal A2
R8	IO_SA_A4	GPIO	1	1st GPIO south bank signal A4
R9	IO_SA_A6	GPIO	1	1st GPIO south bank signal A6
R10	IO_SA_A7	GPIO	1	1st GPIO south bank signal A7

(continued on next page)

Table 5.4: *CCGM1A2 Pin list sorted by ball name*

(continued from previous page)

Ball	Signal name	Signal group	Reset category	Description
R11	IO_SB_B4	GPIO	1	2nd GPIO south bank signal B4
R12	GND	Ground	0	Ground
R13	IO_SB_A5	GPIO	1	1st function: 2nd GPIO south bank signal A5
R13	CLK3	GPIO	1	2nd function: 4th clock input
R14	IO_SB_B5	GPIO	1	2nd GPIO south bank signal B5
R15	VDD_SB	Power	0	2nd GPIO south bank power supply
R16	GND	Ground	0	Ground
R17	IO_SB_A1	GPIO	1	2nd GPIO south bank signal A1
R18	IO_SB_B1	GPIO	1	2nd GPIO south bank signal B1
T1	VDD_WA	Power	0	1st GPIO west bank power supply
T2	JTAG_TDI	GPIO	4	1st function: JTAG data input
T2	IO_WA_A4	GPIO	1	2nd function: 1st GPIO west bank signal A4
T3	JTAG_TMS	GPIO	4	1st function: JTAG test mode select
T3	IO_WA_B4	GPIO	1	2nd function: 1st GPIO west bank signal B4
T4	GND	Ground	0	Ground
T5	CFG_MD1	GPIO	4	1st function: Configuration mode bit 1
T5	IO_WA_B0	GPIO	1	2nd function: 1st GPIO west bank signal B0
T6	IO_SA_B1	GPIO	1	1st GPIO south bank signal B1
T7	IO_SA_B2	GPIO	1	1st GPIO south bank signal B2
T8	IO_SA_B4	GPIO	1	1st GPIO south bank signal B4
T9	IO_SA_B6	GPIO	1	1st GPIO south bank signal B6
T10	IO_SA_B7	GPIO	1	1st GPIO south bank signal B7
T11	GND	Ground	0	Ground
T12	SER_CLK	Other	4	SerDes clock, positive LVDS signal (or single ended)
T13	SER_CLK_N	Other	1	SerDes clock, negative LVDS signal
T14	VDD_CLK	Power	0	Clock signal power supply
T15	RST_N	Other	4	Reset
T16	VDD_SER_PLL	Power	0	SerDes PLL power supply (1.0 V to 1.1 V ±50 mV)
T17	GND	Ground	0	Ground
T18	VDD_SB	Power	0	2nd GPIO south bank power supply
U1	POR_EN	GPIO	2	1st function: Enable power-on reset

(continued on next page)

Table 5.4: *CCGM1A2 Pin list sorted by ball name*

(continued from previous page)

Ball	Signal name	Signal group	Reset category	Description
U1	IO_WA_A3	GPIO	1	2nd function: 1st GPIO west bank signal A3
U2	JTAG_TDO	GPIO	2	1st function: JTAG data output
U2	IO_WA_B3	GPIO	1	2nd function: 1st GPIO west bank signal B3
U3	VDD_WA	Power	0	1st GPIO west bank power supply
U4	CFG_MD2	GPIO	4	1st function: Configuration mode bit 2
U4	IO_WA_A1	GPIO	1	2nd function: 1st GPIO west bank signal A1
U5	IO_SA_A0	GPIO	1	1st GPIO south bank signal A0
U6	VDD_SA	Power	0	1st GPIO south bank power supply
U7	IO_SA_A3	GPIO	1	1st GPIO south bank signal A3
U8	IO_SA_A5	GPIO	1	1st GPIO south bank signal A5
U9	VDD_SA	Power	0	1st GPIO south bank power supply
U10	IO_SA_A8	GPIO	1	1st GPIO south bank signal A8
U11	SER_RX_P0	SerDes	1	Receive data line of interface 0, positive LVDS signal
U12	VDD_SER	Power	0	SerDes core power supply (1.0 V to 1.1 V ±50 mV)
U13	SER_TX_P0	SerDes	1	Transmit data line of interface 0, positive LVDS signal
U14	SER_RX_P1	SerDes	1	Receive data line of interface 1, positive LVDS signal
U15	GND	Ground	0	Ground
U16	SER_TX_P1	SerDes	1	Transmit data line of interface 1, positive LVDS signal
U17	IO_SB_A0	GPIO	1	2nd GPIO south bank signal A0
U18	IO_SB_B0	GPIO	1	2nd GPIO south bank signal B0
V1	GND	Ground	0	Ground
V2	CFG_FAILED_N	GPIO	2	1st function: Configuration failed signal
V2	IO_WA_A2	GPIO	1	2nd function: 1st GPIO west bank signal A2
V3	CFG_DONE	GPIO	2	1st function: Configuration done signal
V3	IO_WA_B2	GPIO	1	2nd function: 1st GPIO west bank signal B2
V4	CFG_MD3	GPIO	4	1st function: Configuration mode bit 3
V4	IO_WA_B1	GPIO	1	2nd function: 1st GPIO west bank signal B1
V5	IO_SA_B0	GPIO	1	1st GPIO south bank signal B0
V6	GND	Ground	0	Ground

(continued on next page)

Table 5.4: *CCGM1A2 Pin list sorted by ball name*

(continued from previous page)

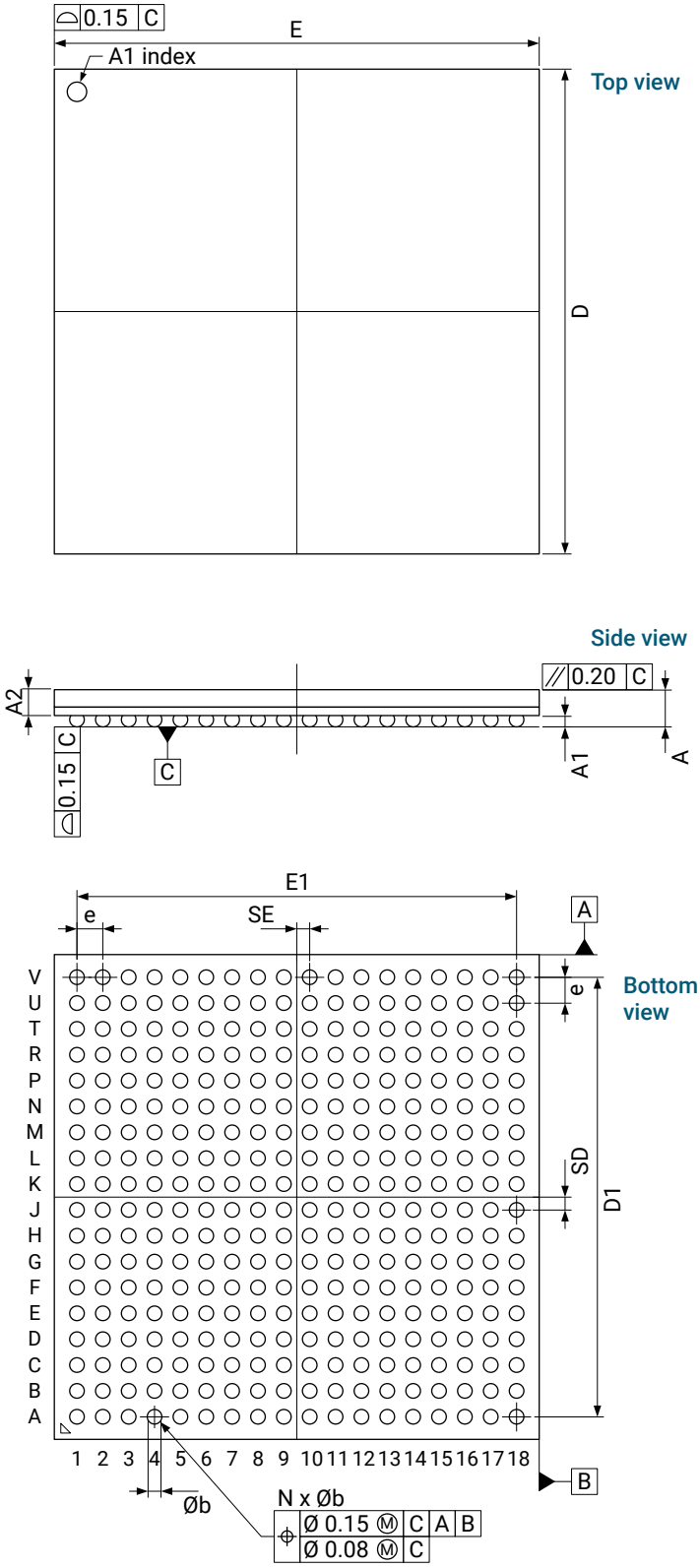
Ball	Signal name	Signal group	Reset category	Description
V7	IO_SA_B3	GPIO	1	1st GPIO south bank signal B3
V8	IO_SA_B5	GPIO	1	1st GPIO south bank signal B5
V9	GND	Ground	0	Ground
V10	IO_SA_B8	GPIO	1	1st GPIO south bank signal B8
V11	SER_RX_N0	SerDes	1	Receive data line of interface 0, negative LVDS signal
V12	SER_RTERM0	SerDes	0	Line termination of interface 0
V13	SER_TX_N0	SerDes	1	Transmit data line of interface 0, negative LVDS signal
V14	SER_RX_N1	SerDes	1	Receive data line of interface 1, negative LVDS signal
V15	SER_RTERM1	SerDes	0	Line termination of interface 1
V16	SER_TX_N1	SerDes	1	Transmit data line of interface 1, negative LVDS signal
V17	VDD_SER	Power	0	SerDes core power supply (1.0 V to 1.1 V ±50 mV)
V18	GND	Ground	0	Ground

Reset categories:

- **0:** No reset (supply or analog input pin)
- **1:** Pin disabled, high-z
- **2:** Pin characteristic already during reset state
- **3:** Pin characteristic depends on the configuration mode (SPI master: I/O, else: input)
- **4:** Input pin

Chapter 6

Mechanical Dimensions



CCGM1A1			
Ref.	Dimensions in mm		
	Min	Nom	Max
A	1.080	1.190	1.300
A1	0.300	0.350	0.400
A2	0.780	0.840	0.900
Øb	0.400	0.450	0.500
D	14.900	15.000	15.100
E	14.900	15.000	15.100
D1	13.600 BSC		
E1	13.600 BSC		
SE	0.400 BSC		
SD	0.400 BSC		
e	0.800 BSC		

CCGM1A2			
Ref.	Dimensions in mm		
	Min	Nom	Max
A	1.680	--	1.880
A1	0.311	0.361	0.411
A2	1.357	1.420	1.483
Øb	0.400	0.450	0.500
D	14.900	15.000	15.100
E	14.900	15.000	15.100
D1	13.600 BSC		
E1	13.600 BSC		
SE	0.400 BSC		
SD	0.400 BSC		
e	0.800 BSC		

- All dimensions and tolerances are conform to ASME Y14.5M-1994.
- Compliant to JEDEC registered outline MO-275.
- Ultra low alpha compound.

Figure 6.1: *FATC FBGA 324 balls package dimensions*

Chapter 7

Soldering Guidelines

The recommended profile for soldering reflow of CCGM1A1 / CCGM1A2 for Pb-free assembly correspond to the commonly applied JEDEC Standard JSTD-020E. Table 7.1 and Figure 7.1 illustrate the soldering reflow.

CCGM1A1 / CCGM1A2 may have a crack when thermal stress is applied during assembly if it has absorbed atmospheric moisture. It is recommended that the chip is handled under specific conditions described as follows:

Storage Condition after unpacking (as maximum): Within 168 hours, 30 °C / 60 % RH (MSL 3)

Rebake Condition (as minimum): 125 °C for 24 hours

Table 7.1: *Pb-free reflow soldering profile*

Symbol	Description	Parameter value
Preheat:		
T_{SMIN}	Minimum temperature	150 °C
T_{SMAX}	Maximum temperature	200 °C
t_S	Time between T_{SMIN} and T_{SMAX}	60 . . 180 s
Ramp-up:		
T_L	Liquidous temperature	217 °C
	Ramp-up rate T_{SMAX} to T_L	max 3 °C/s
t_L	Time maintained above T_L	60 . . 150 s
T_P	Peak temperature	260 + 0 / − 5 °C
	Time from 25 °C to peak temperature	max 8 min
t_P	Time within 5 °C of peak temperature	20 . . 40 s
Ramp-down:		
	Ramp-down rate	max 6 °C/s

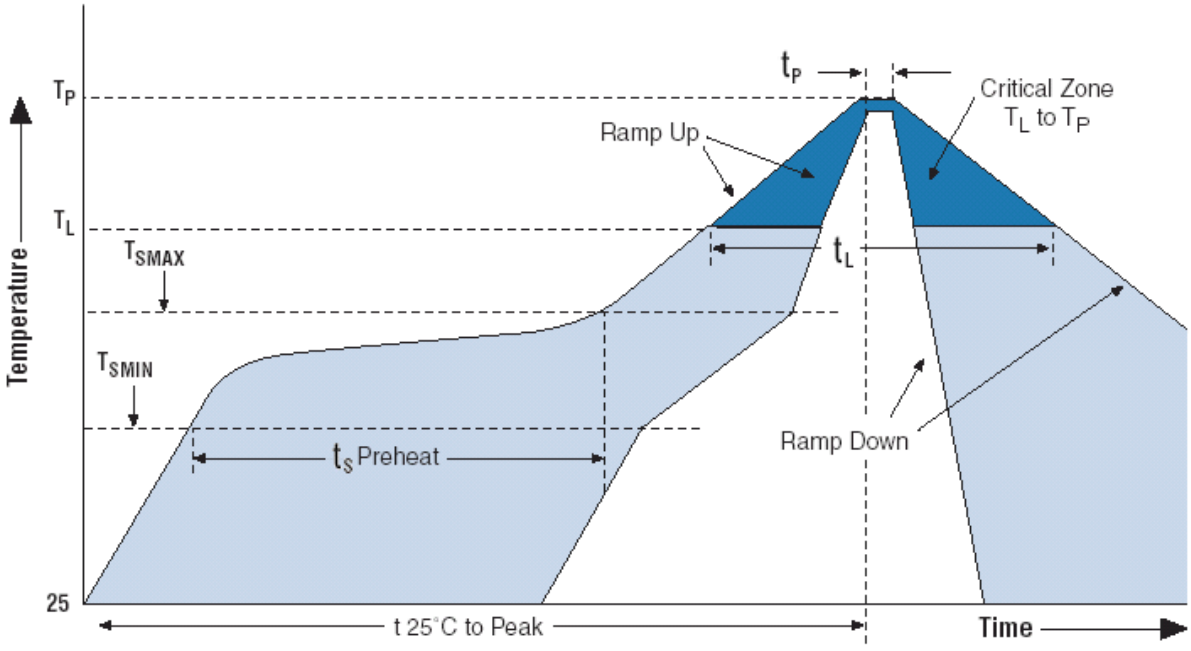


Figure 7.1: *Reflow soldering profile*

GateMate™ FPGA Datasheet
CCGM1A1
CCGM1A2

DS1001
February 2025

Cologne Chip