V. 12/23

GateMate™ FPGA Datasheet

CCGM1A1

DS1001

Datasheet December 2023

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^{TM} FPGA Series and is part of the GateMate^{TM} documentation collection.

For more information please refer to the following documents:

- Technology Brief of GateMate^{TM} FPGA
- **DS1002** – GateMate^{TM} FPGA Programmer Board Ver. 1.3 Datasheet
- **DS1003** – GateMate^{TM} FPGA Evaluation Board Ver. 3.1 Datasheet
- **UG1001** – GateMate^{TM} FPGA Primitives Library
- **UG1002** – GateMate^{TM} FPGA Toolchain Installation User Guide

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

Revision History

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

DS1001 – GateMate^{TM} FPGA CCGM1A1 Datasheet

Date	Remarks
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 74.
November 2023	Information added in Section 4.
January 2023	Minor changes in Section 3.1.
December 2022	• Section 2.6 (Global Mesh Architecture) from page 46 added. • Section 2.7 (Clocking Schemes) from page 52 added. • `SPI_CLK` (pin `N4`) and `JTAG_TCK` (pin `R3`) are no longer available as dedicated clock input pins. • Chapter 7 (Soldering Guidelines) added. • Figure 2.2 on page 22 extended by details.
May 2022	• Sections 3.1.2 (Synthesis), 3.1.3 (Implementation) and 3.1.4 (Configuration) from page 67 added. • Additional explanation to the power-on reset module: – Section 3.2.1 from page 74 expanded. – Sections 3.2.2 and 3.2.2 from page 75 added. – Reset characteristics in Table 4.6 on page 98 added. • Clock pins renamed from `CLK[4:1]` to `CLK[3:0]` (BGA pins `N14`, `P12`, `P14` and `R13`).
February 2022	Updated workflow description in Section 3.1.1 from page 65 with Yosys support added.
October 2021	Additional explanation to Section 2.4.8 on page 36.
July 2021	• Expanded Block RAM documentation in Section 2.4 from page 27. • Updated maximum PLL output frequencies in Section 2.5 on page 45.
...	
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 Section 5.2 on page 101 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^{TM} 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^{TM} 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

- Novel programmable element architecture
  - 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
  - I/O voltage range from 1.2 to 2.5 V
- 4 clock generators (PLL)
  - Maximum PLL output frequency from 250 MHz to 416.75 MHz
- 2.5 Gb/s serializer / deserializer (SerDes) controller
- Flexible memory resources
  - 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
- Application 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

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 2.8.

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.

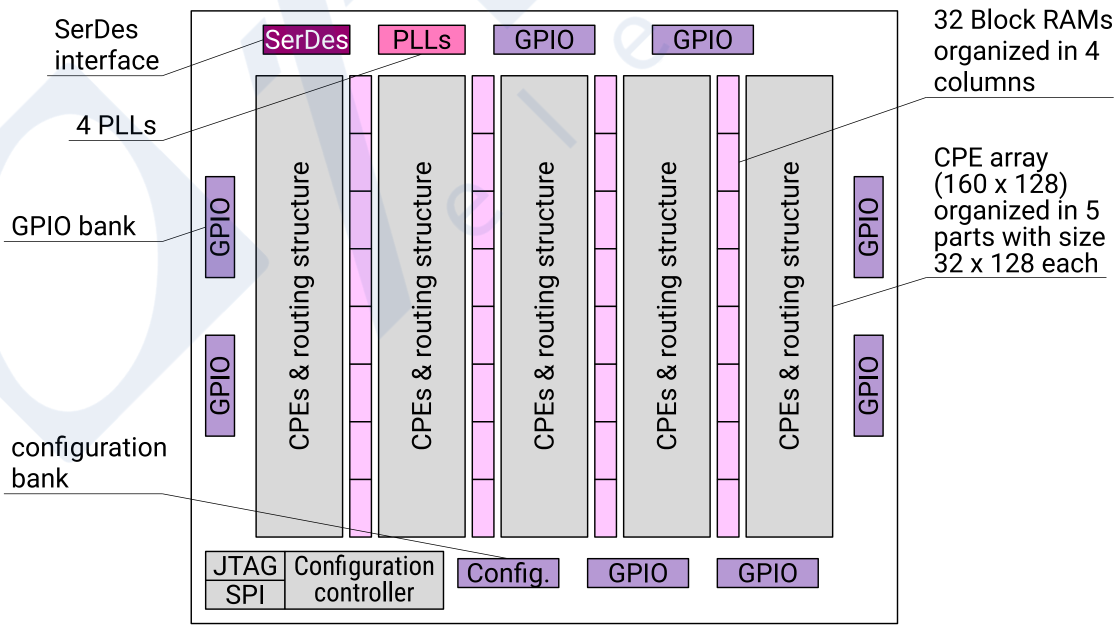


Figure 2.1: *Simplified architecture overview*

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 $\times$ 128 matrix. 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 $\times$ 2-bit multiplier, expandable to any multiplier size

Figure 2.2 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^{TM} 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.

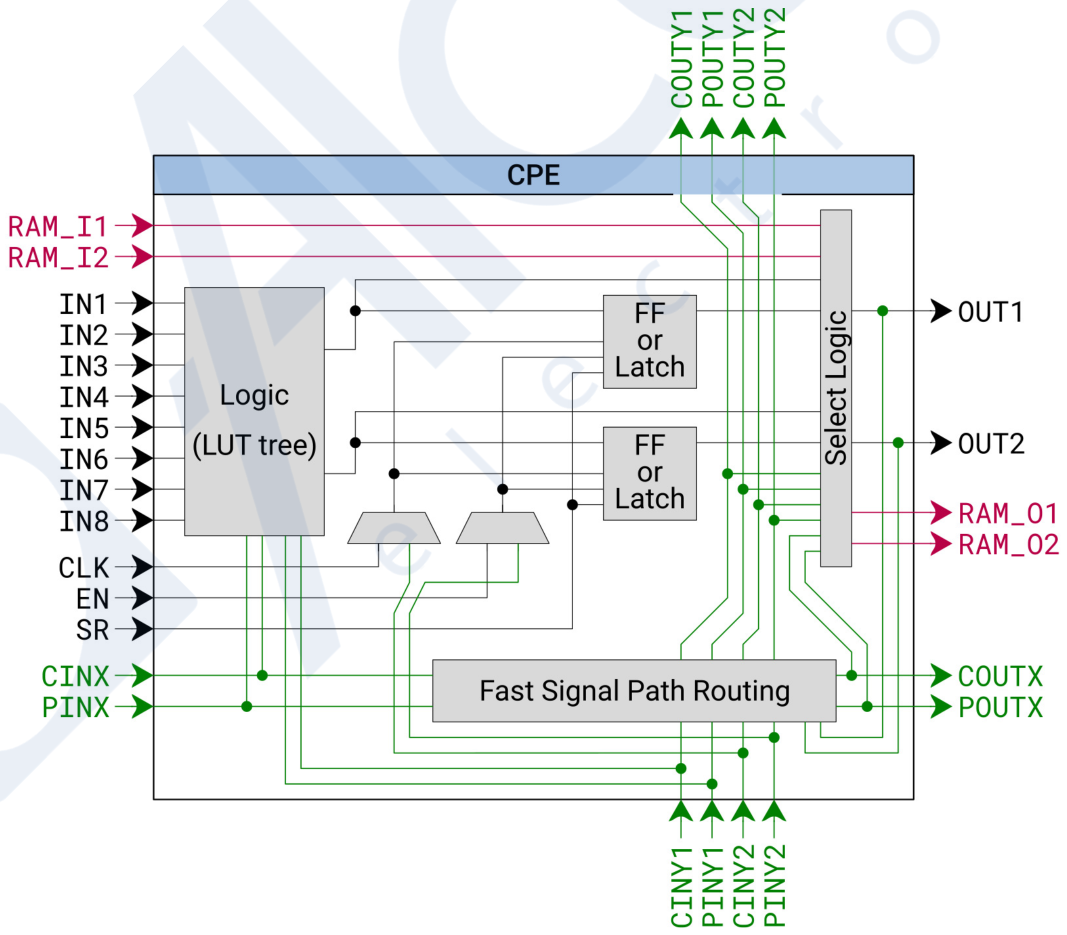


Figure 2.2: *Cologne Programmable Element (CPE)*

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^{TM} FPGA offers 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 2.3 and 2.4 show the structure of the pad cells and their interconnection to the FPGA elements.

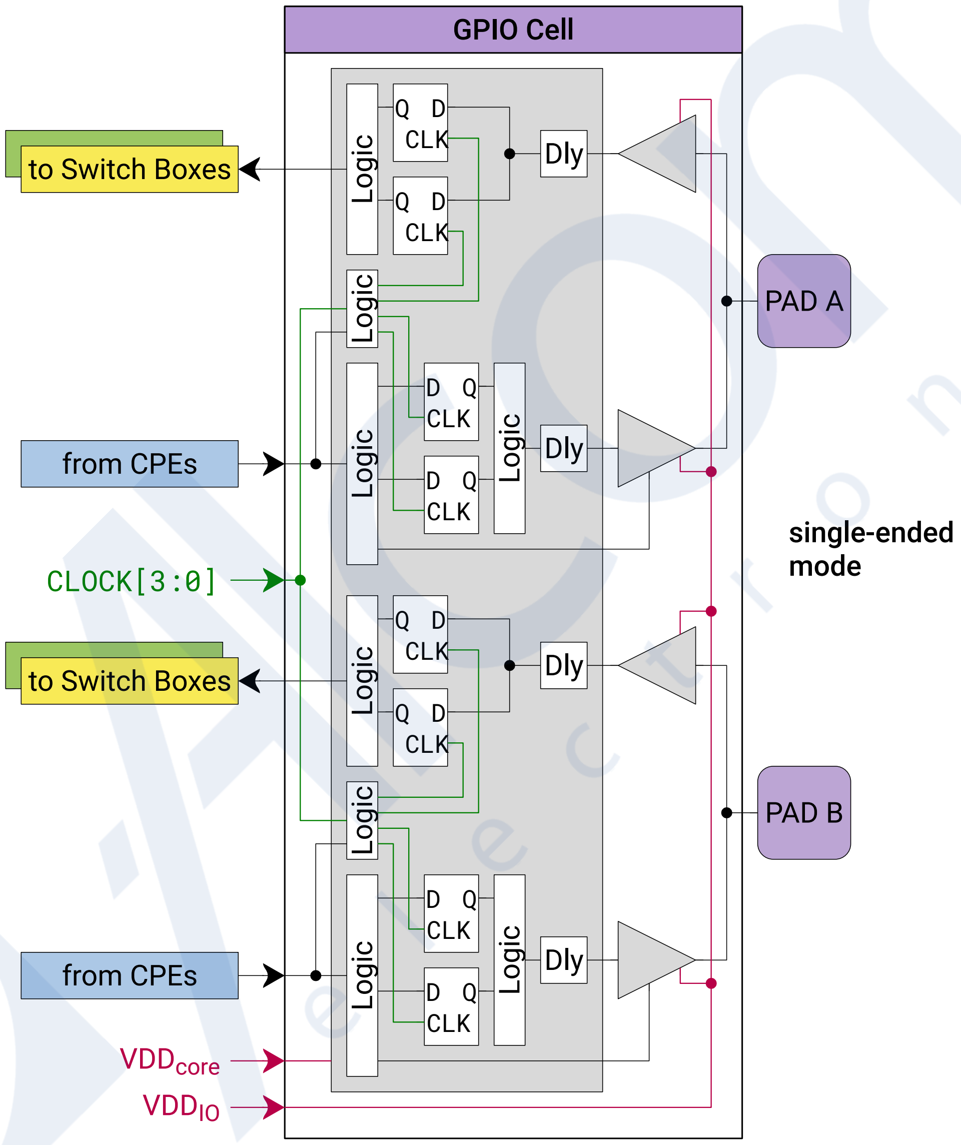


Figure 2.3: *GPIO cell in single-ended mode*

Nine pad cells build up a GPIO bank as shown in Figure 2.5 on page 26. 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 standards JESD8-5 (2.5 V) and JESD8-7 (1.8 V and 1.2 V).

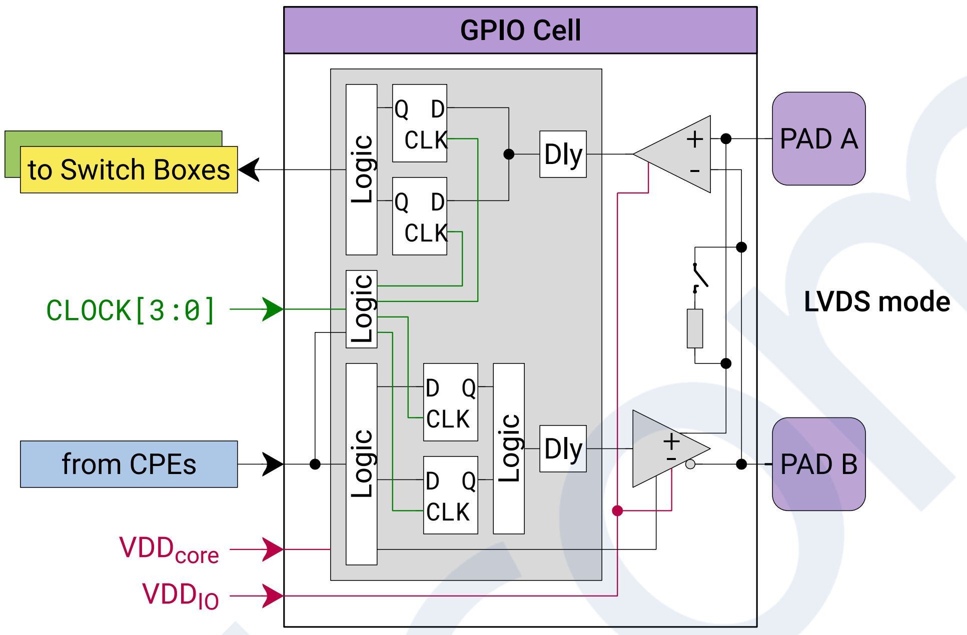


Figure 2.4: *GPIO cell in LVDS mode*

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
- 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
- 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.

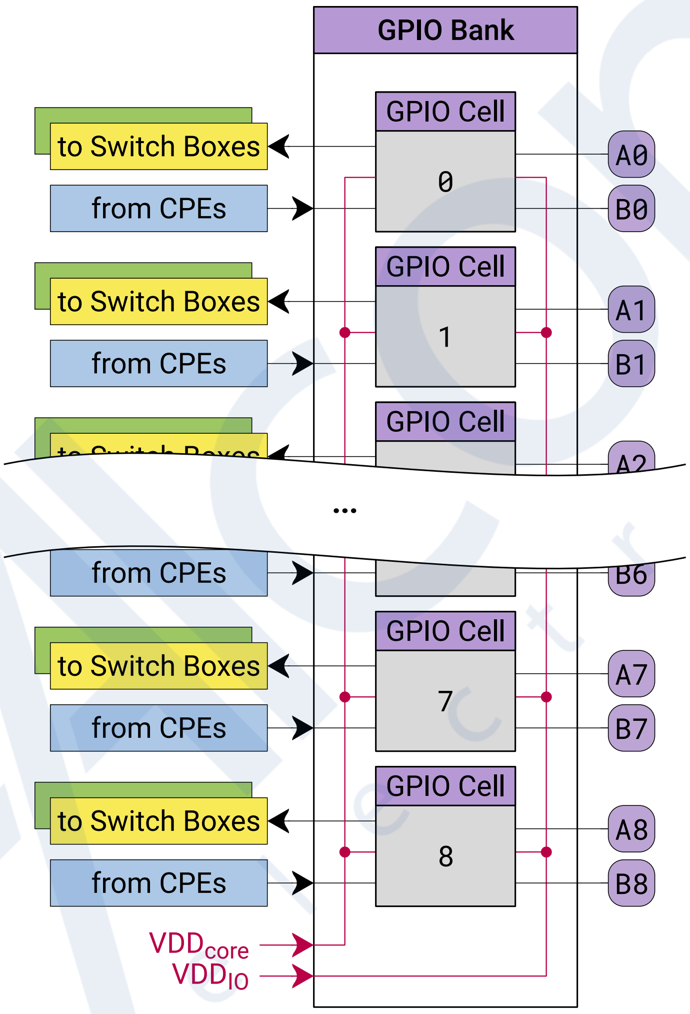


Figure 2.5: *GPIO bank*

2.4 Embedded Block RAM

2.4.1 Block RAM Overview

The GateMate™ FPGA 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 2.6. Block RAMs have an address (x, y) with $x = 0..3$ and $y = 0..7$.

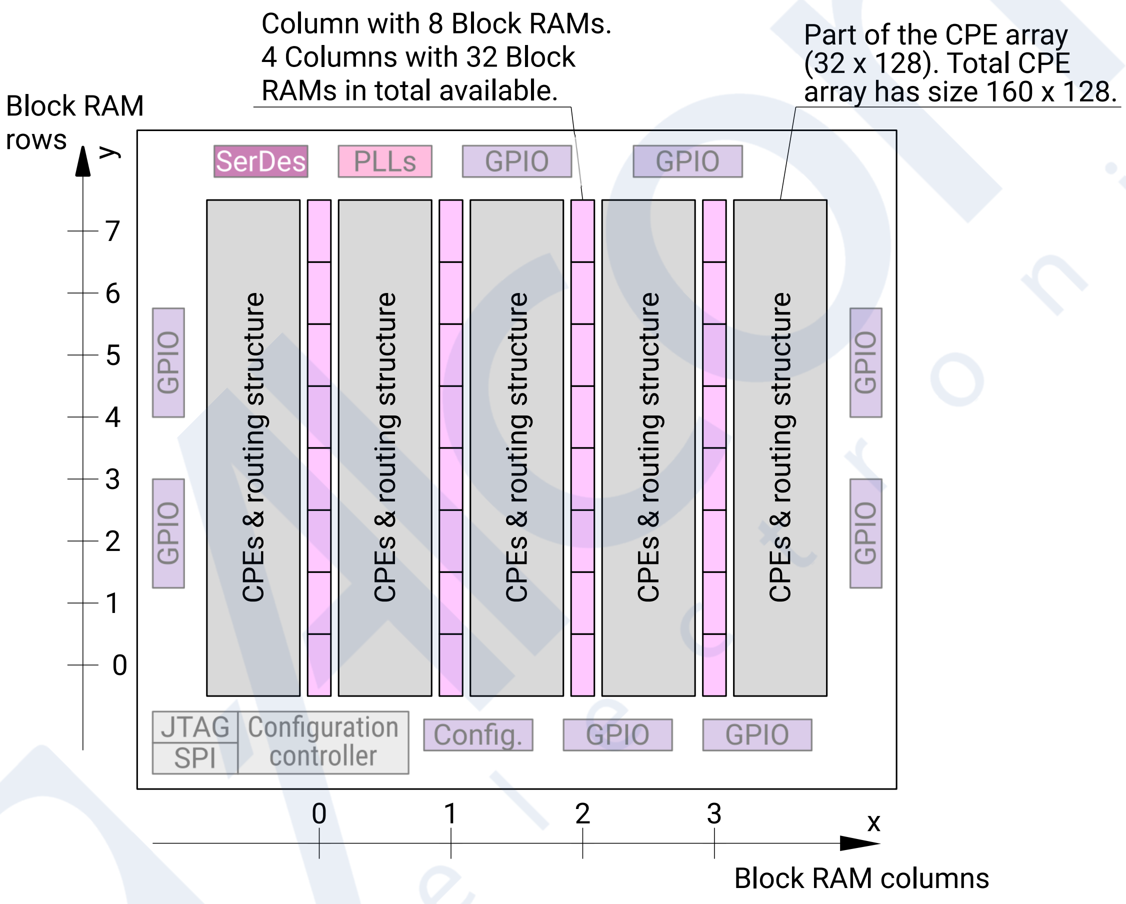


Figure 2.6: *Block RAM arrangement*

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 errors 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 2.1 and 2.2 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 2.1: *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 2.2: *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 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 2.7 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.

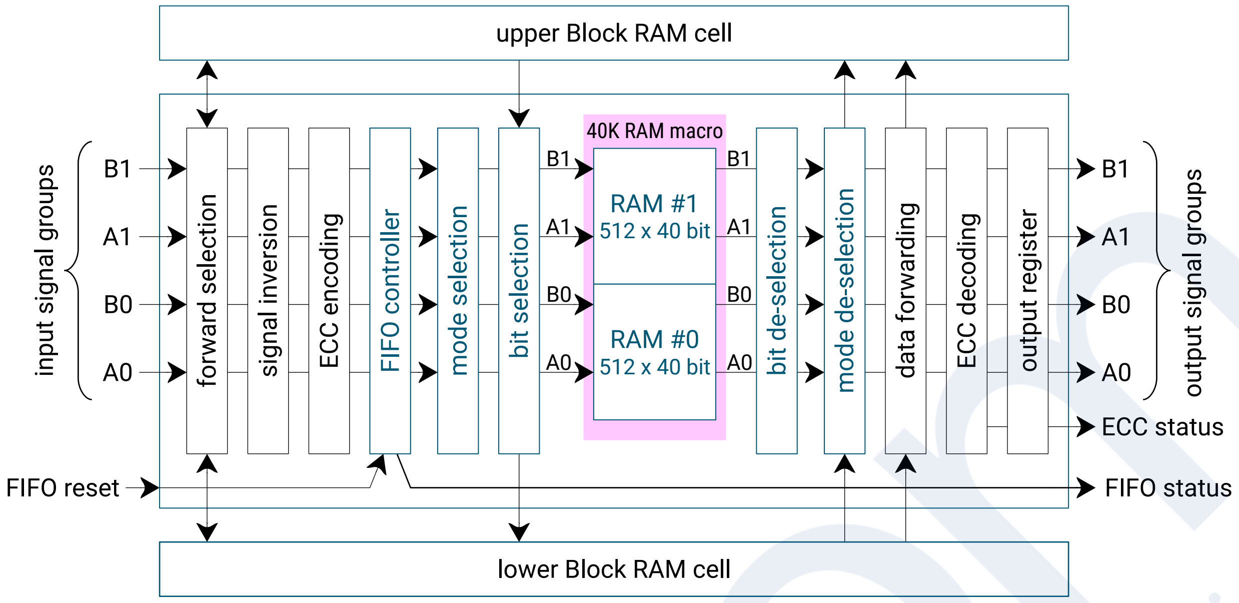


Figure 2.7: *High level block diagram of a block RAM cell*

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 2.3: *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 incoming 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 2.1 and 2.2. 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 2.8 and 2.9 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.

Table 2.4 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 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

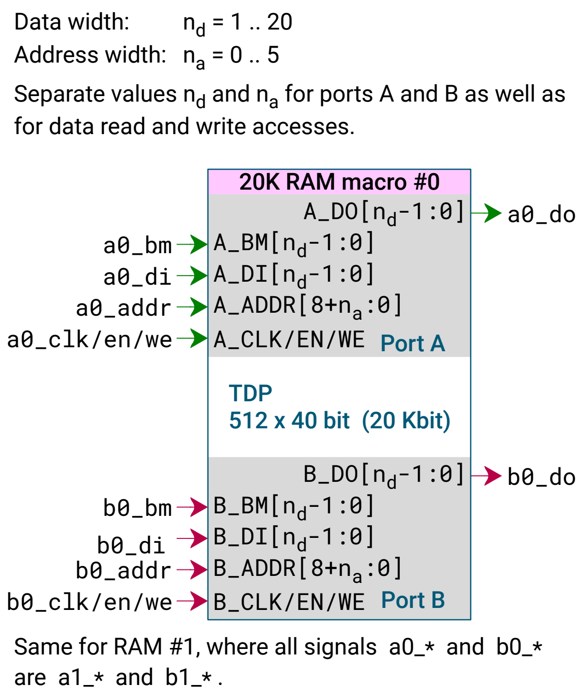


Figure 2.8: *Internal data flow of DPSRAM in TDP 20K mode*

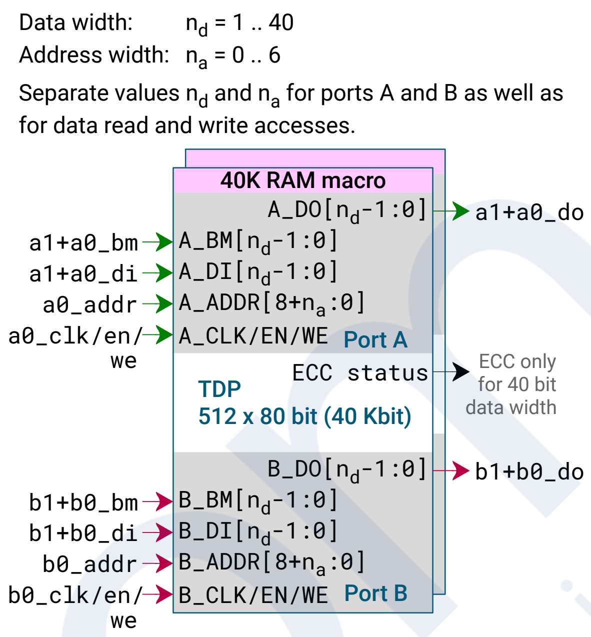


Figure 2.9: *Internal data flow of DPSRAM in TDP 40K mode*

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.

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.

Table 2.4: *Pin wiring in TDP 20K mode*

Width	Depth	Address-in bus [1]	Data-in bus [1]	Data-out bus [1]
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.

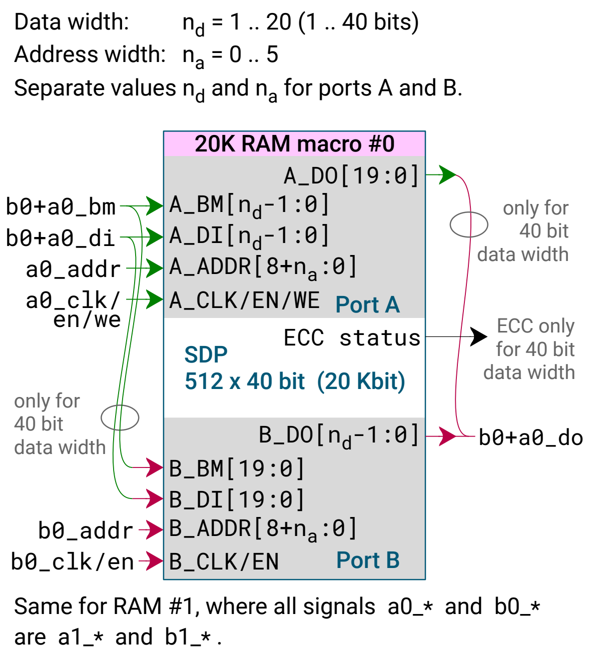

Data width: n_d = 1 .. 20 (1 .. 40 bits)
Address width: n_a = 0 .. 5
Separate values n_d and n_a for ports A and B.
20K RAM macro #0
A_DO[19:0]
b0+a0_bm
A_BM[n_d-1:0]
b0+a0_di
A_DI[n_d-1:0]
a0_addr
A_ADDR[8+n_a:0]
a0_clk/en/we
A_CLK/EN/WE
Port A
only for 40 bit data width
ECC status
ECC only for 40 bit data width
SDP
512 x 40 bit (20 Kbit)
only for 40 bit data width
B_DO[n_d-1:0]
b0+a0_do
B_BM[19:0]
B_DI[19:0]
b0_addr
B_ADDR[8+n_a:0]
b0_clk/en
B_CLK/EN
Port B
Same for RAM #1, where all signals a0_* and b0_* are a1_* and b1_*.

Figure 2.10: *Internal data flow of DPSRAM in SDP 20K mode*

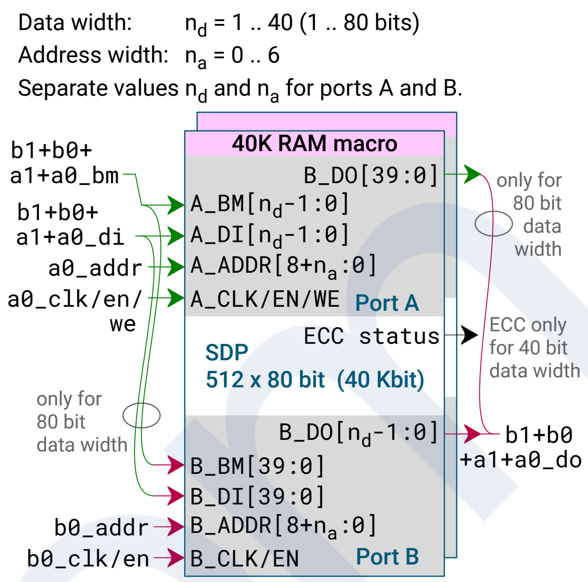


Figure 2.11: *Internal data flow of DPSRAM in SDP 40K mode*

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 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 2.5: *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 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.

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 2.12 on page 34. By that, two adjacent 32K $\times$ 1 bit Block RAM cells can be combined to form a 64K $\times$ 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 $\times$ 1 bit TDP mode. Bit 0 of the address bus selects the upper or lower Block RAM cell for the write or read operation.

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

Table 2.6: *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 2.7: *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] Data width of 80 bits only supported in 40K SDP mode (see Tables 2.1 and 2.2).

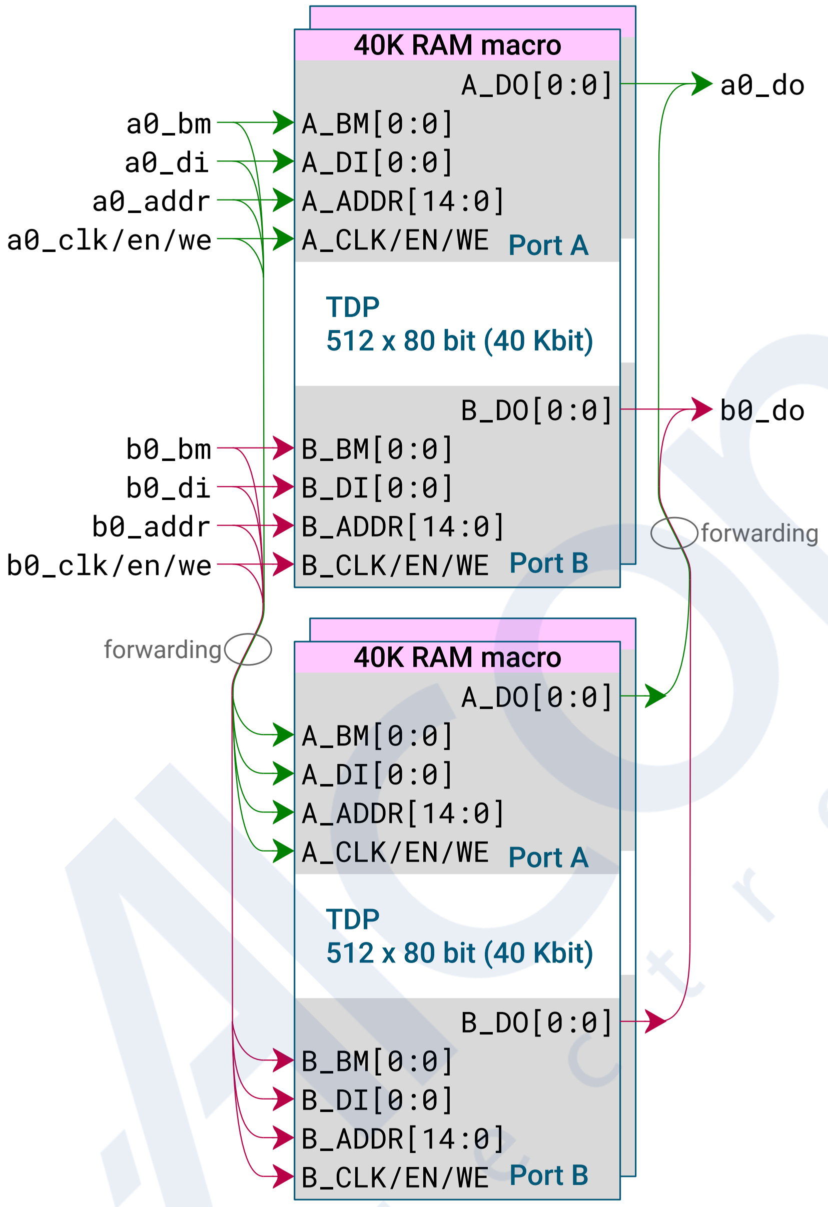


Figure 2.12: *Structure of two adjacent DPSRAMs forming one 64K × 1 bit memory via cascade feature*

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 2.13 on page 35. The scheme shown here can be applied to both SDP and TDP modes in 20K as well as in 40K configurations.

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

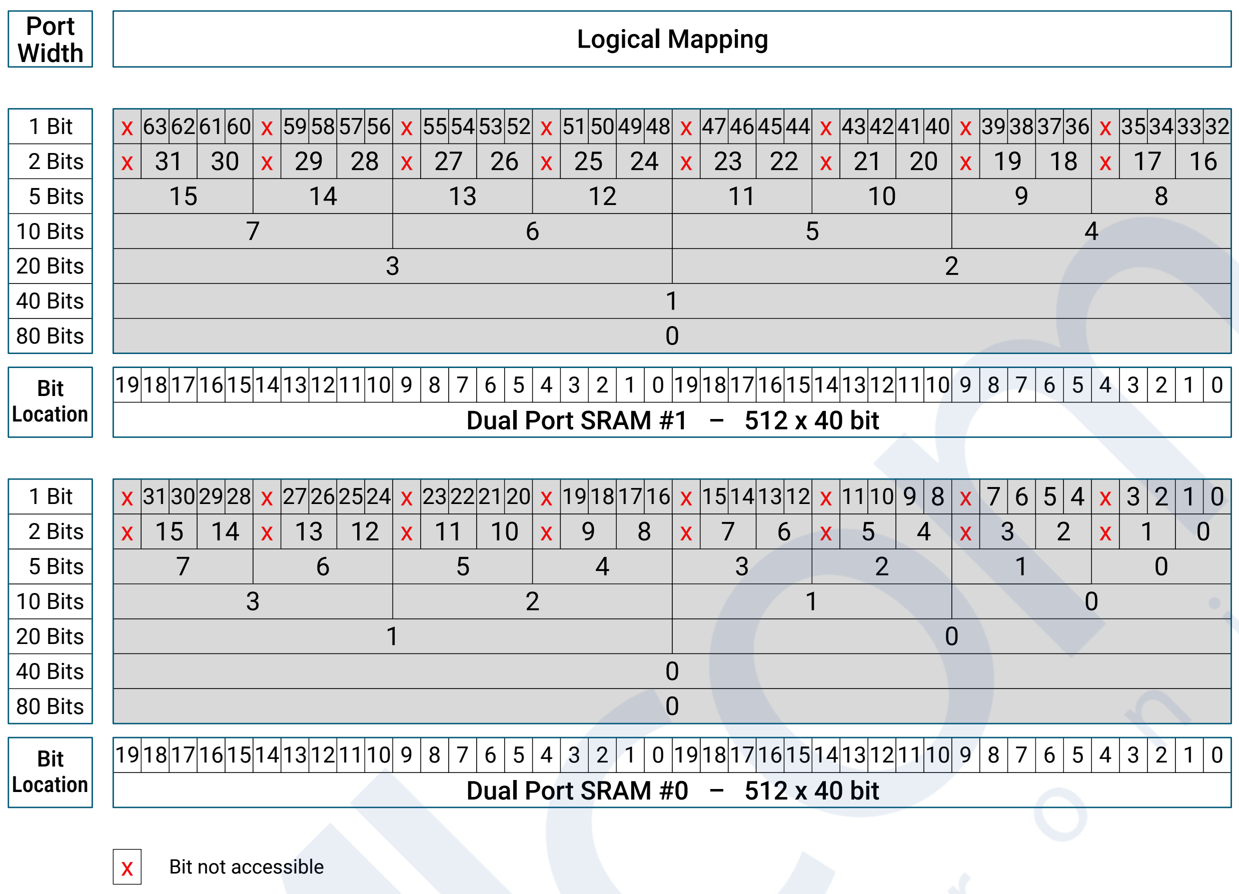


Figure 2.13: *Logical data mapping of memory at physical address 0*

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 2.8 and 2.9. 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 2.8: *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 2.9: *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 2.14.

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.

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 2.15.

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.

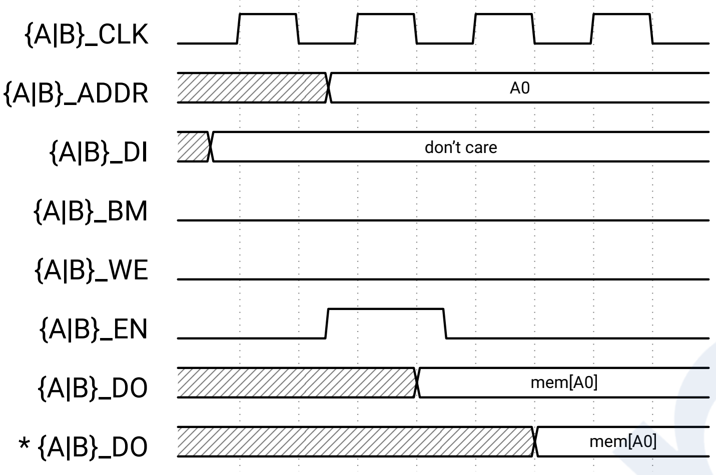


Figure 2.14: *Timing diagram of a single read access with optional (*) output register*

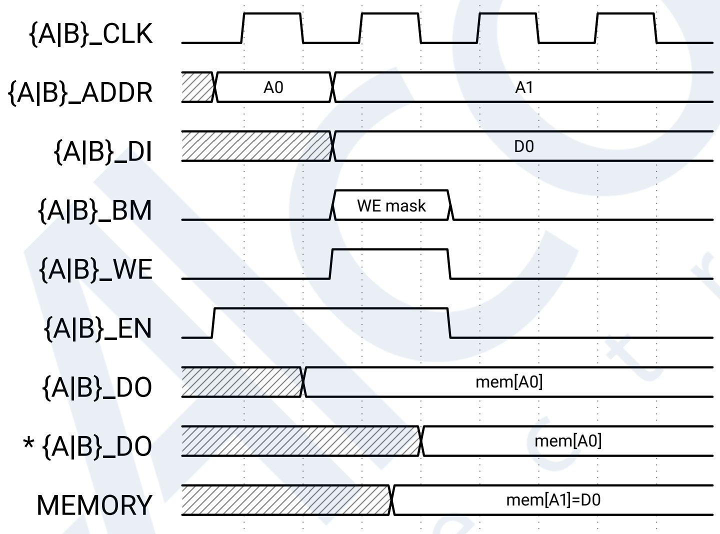


Figure 2.15: *Timing diagram of a NO CHANGE access with optional (*) output register*

The timing diagram 2.16 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.

2.4.9 FIFO Application

Each GateMate^{TM} 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 syn-

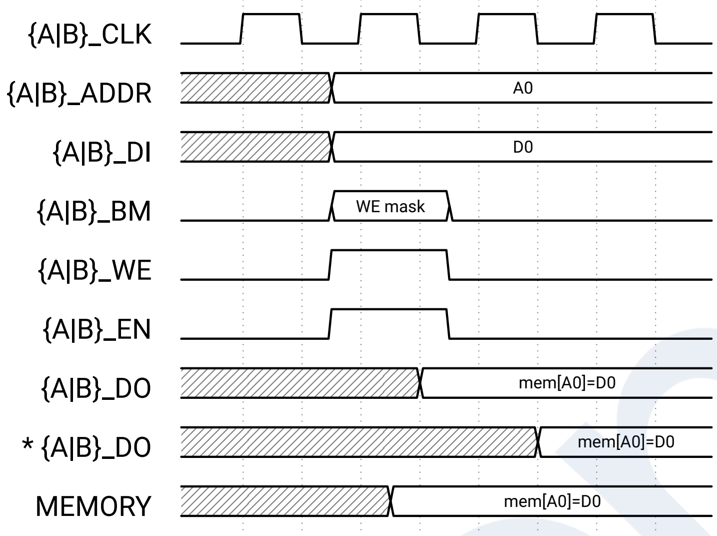


Figure 2.16: *Timing diagram of a* `WRITE THROUGH` *access with optional (*) output register*

chronous 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 2.5 (page 32) and 2.7. 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

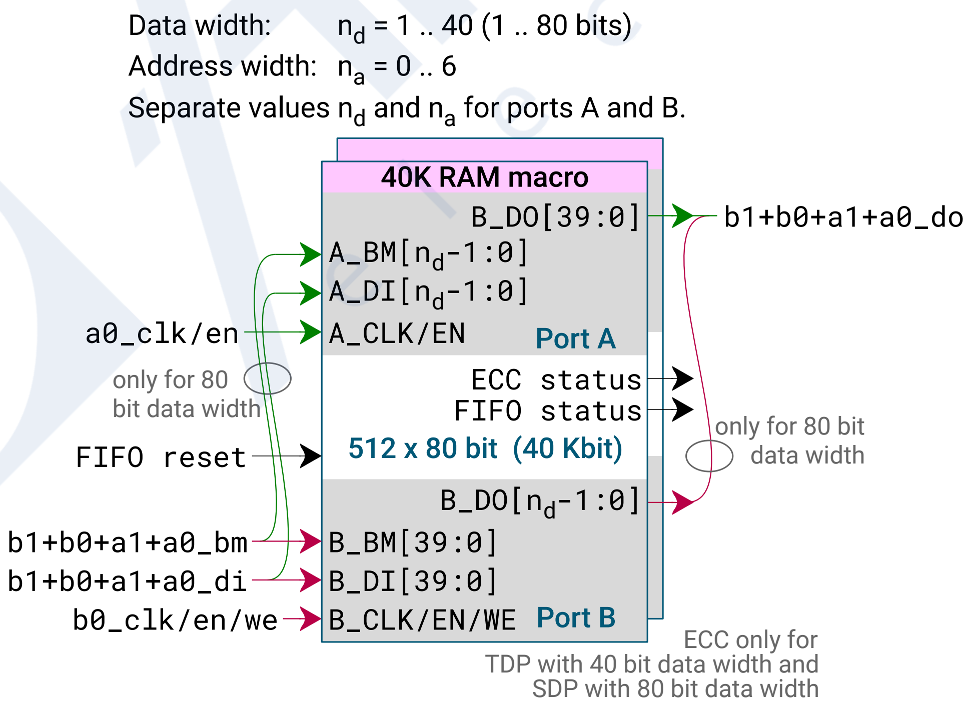

Data width: n_d = 1 .. 40 (1 .. 80 bits)
Address width: n_a = 0 .. 6
Separate values n_d and n_a for ports A and B.
40K RAM macro
B_DO[39:0]
b1+b0+a1+a0_do
A_BM[n_d-1:0]
A_DI[n_d-1:0]
a0_clk/en
A_CLK/EN
Port A
only for 80 bit data width
ECC status
FIFO status
FIFO reset
512 x 80 bit (40 Kbit)
only for 80 bit data width
B_DO[n_d-1:0]
b1+b0+a1+a0_bm
B_BM[39:0]
b1+b0+a1+a0_di
B_DI[39:0]
b0_clk/en/we
B_CLK/EN/WE
Port B
ECC only for TDP with 40 bit data width and SDP with 80 bit data width

Figure 2.17: *Internal data flow of DPSRAM in FIFO mode*

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 2.10. 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 a 13 bit register during configuration or it is set dynamically using the inputs F_ALMOST_FULL_OFFSET and F_ALMOST_EMPTY_OFFSET.

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

Table 2.10: *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 2.11 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.

Table 2.11: *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 2.12 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 2.12: *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 2.18 illustrates the writing to an empty synchronous FIFO.

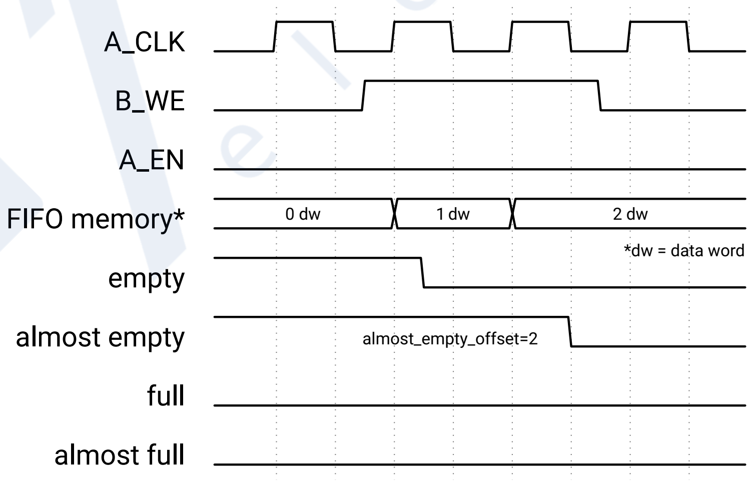


Figure 2.18: *Writing to an empty synchronous FIFO*

The timing diagram in Figure 2.19 illustrates the writing to an almost full synchronous FIFO.

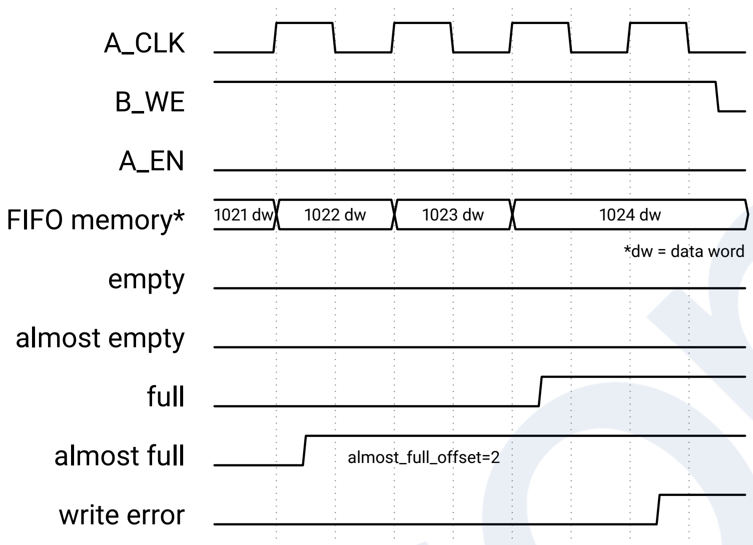


Figure 2.19: *Writing to an almost full synchronous FIFO*

The timing diagram in Figure 2.20 illustrates the reading from a full synchronous FIFO.

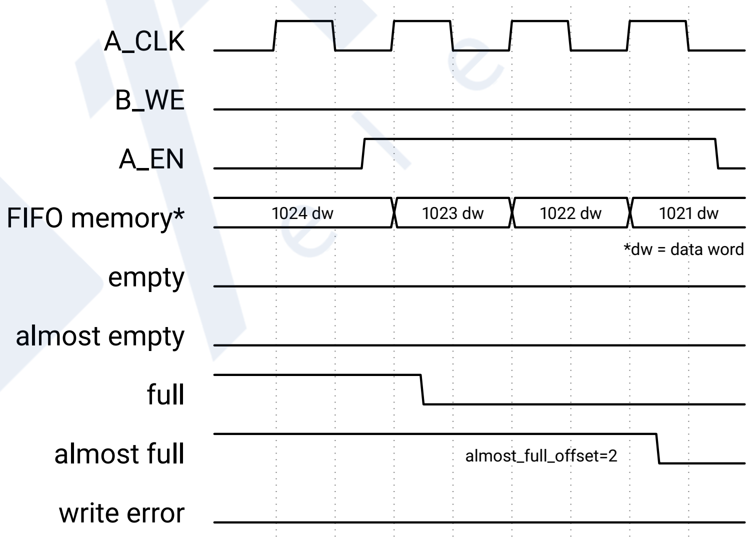


Figure 2.20: *Reading from a full synchronous FIFO*

The timing diagram in Figure 2.21 illustrates the reading from an almost empty synchronous FIFO.

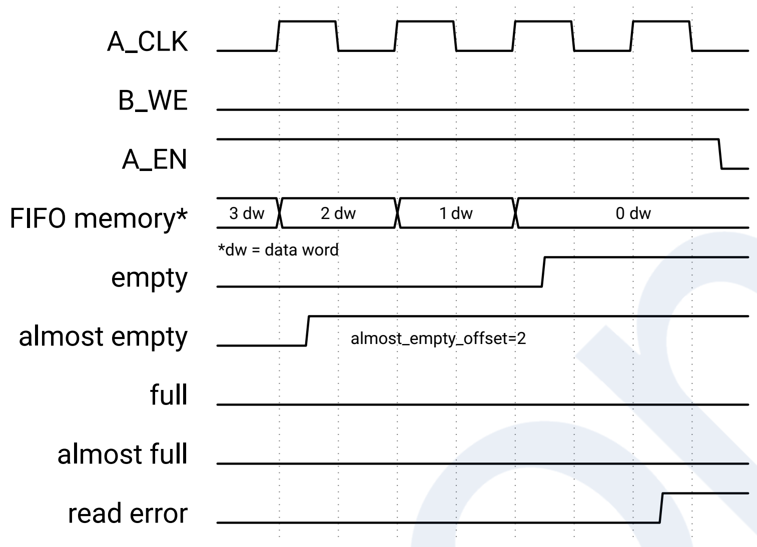


Figure 2.21: *Reading from an almost empty synchronous FIFO*

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 2.22 to 2.25, 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 2.22 illustrates the writing to an empty asynchronous FIFO.

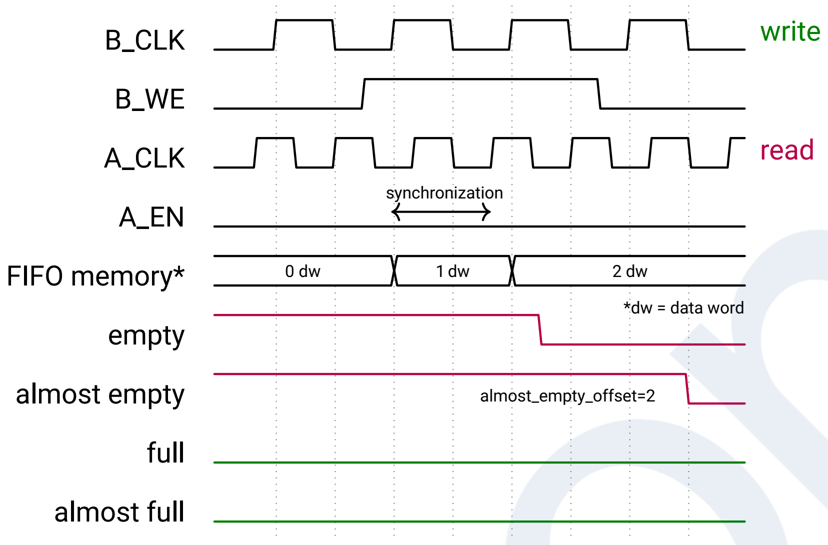


Figure 2.22: *Writing to an empty asynchronous FIFO*

The timing diagram in Figure 2.23 illustrates the writing to an almost full asynchronous FIFO.

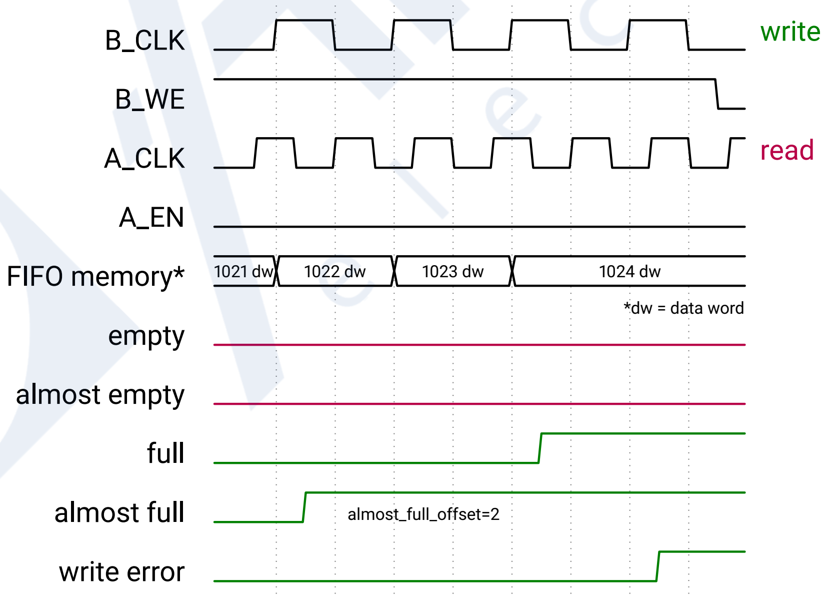


Figure 2.23: *Writing to an almost full asynchronous FIFO*

The timing diagram in Figure 2.24 illustrates the reading from a full asynchronous FIFO.

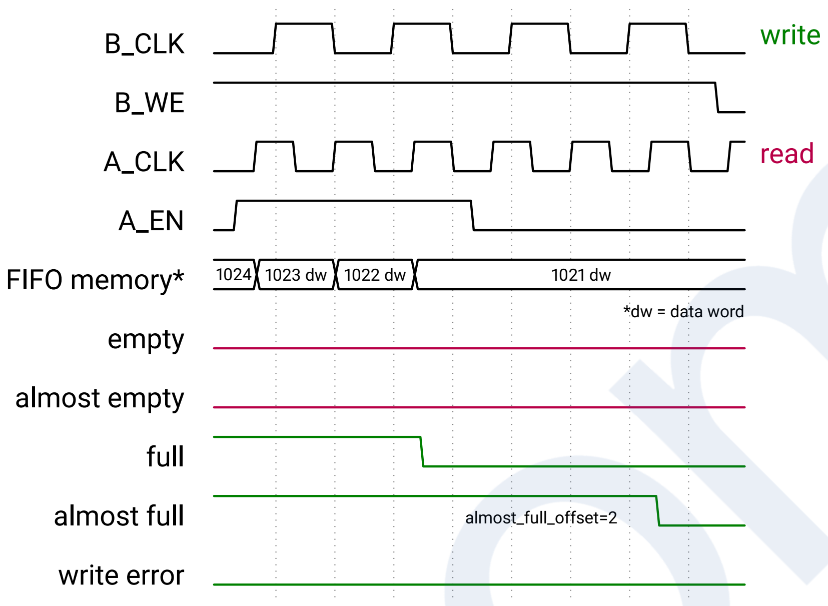


Figure 2.24: *Reading from a full asynchronous FIFO*

The timing diagram in Figure 2.25 illustrates the reading from a full asynchronous FIFO.

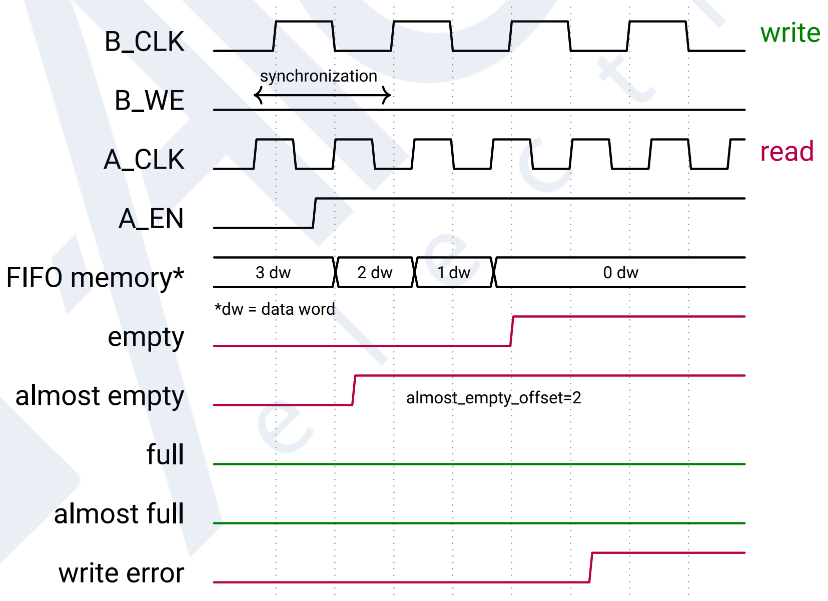


Figure 2.25: *Reading from an almost empty asynchronous FIFO*

2.5 Clock Generators (PLLs)

Four independent PLL clock generators are available:

- All-digital phase-locked loop (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 core clock outputs, enabling wide output and reference frequency ranges.
- Core clock outputs available at four 90°-spaced phases.
- Fast lock-in by binary frequency search.
- Autonomous free-running oscillator mode for FPGA configuration clock after power-up.

The maximum oscillator frequency depends on the FPGA supply voltage as follows:

Low power mode:	$(\mathrm{VDD_{PLL}} = 0.9\,\mathrm{V} \pm 50\,\mathrm{mV})$:	1.000 MHz
Economy mode:	$(\mathrm{VDD_{PLL}} = 1.0\,\mathrm{V} \pm 50\,\mathrm{mV})$:	2.000 MHz
Speed mode:	$(\mathrm{VDD_{PLL}} = 1.1\,\mathrm{V} \pm 50\,\mathrm{mV})$:	2.500 MHz

The maximum PLL output frequency also depends on the FPGA supply voltage as follows:

Low power mode:	$(\mathrm{VDD_{PLL}} = 0.9\,\mathrm{V} \pm 50\,\mathrm{mV})$:	250 MHz
Economy mode:	$(\mathrm{VDD_{PLL}} = 1.0\,\mathrm{V} \pm 50\,\mathrm{mV})$:	312.5 MHz
Speed mode:	$(\mathrm{VDD_{PLL}} = 1.1\,\mathrm{V} \pm 50\,\mathrm{mV})$:	416.75 MHz

There are different reference clock pins selectable individually for each PLL:

- SerDes clock input `SER_CLK` (pin `T12`, can also be used as general purpose clock input)
- 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`)

Reference clocks can be used simultaneously by several PLLs.

2.6 Global Mesh Architecture

2.6.1 Overview of the Global Mesh Signal Injection

The Global Mesh consists of four signal traces which 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 2.26 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.

2.6.2 Clock Input Multiplexers `CLKIN`

The GateMate FPGA has four dedicated clock input pins `CLK0..CLK3`. These clock pins are the second pin functions of GPIO pins (see list on page 45). Further, `SER_CLK` can be used to feed-in a clock signal to the FPGA.

From these five input clocks the `CLKIN` module selects four output clocks as shown in Figure 2.27. Clock inversion (180° phasing) is configurable to any `CLKIN` clock output `PCLK[3:0]`. These clocks are forwarded to the PLLs.

2.6.3 PLL Wrapper

There are four PLL wrappers, each with a PLL as described in Section 2.5. Every clock ouput from the `CLKIN` multiplexers can be used as reference clock of a PLL. Figure 2.28 shows that these `PCLK[3:0]` clocks and alternative user clocks `USR_PCLK[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 2.28 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 wrapper.

Table 2.13: *USR_PCLK⟨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`

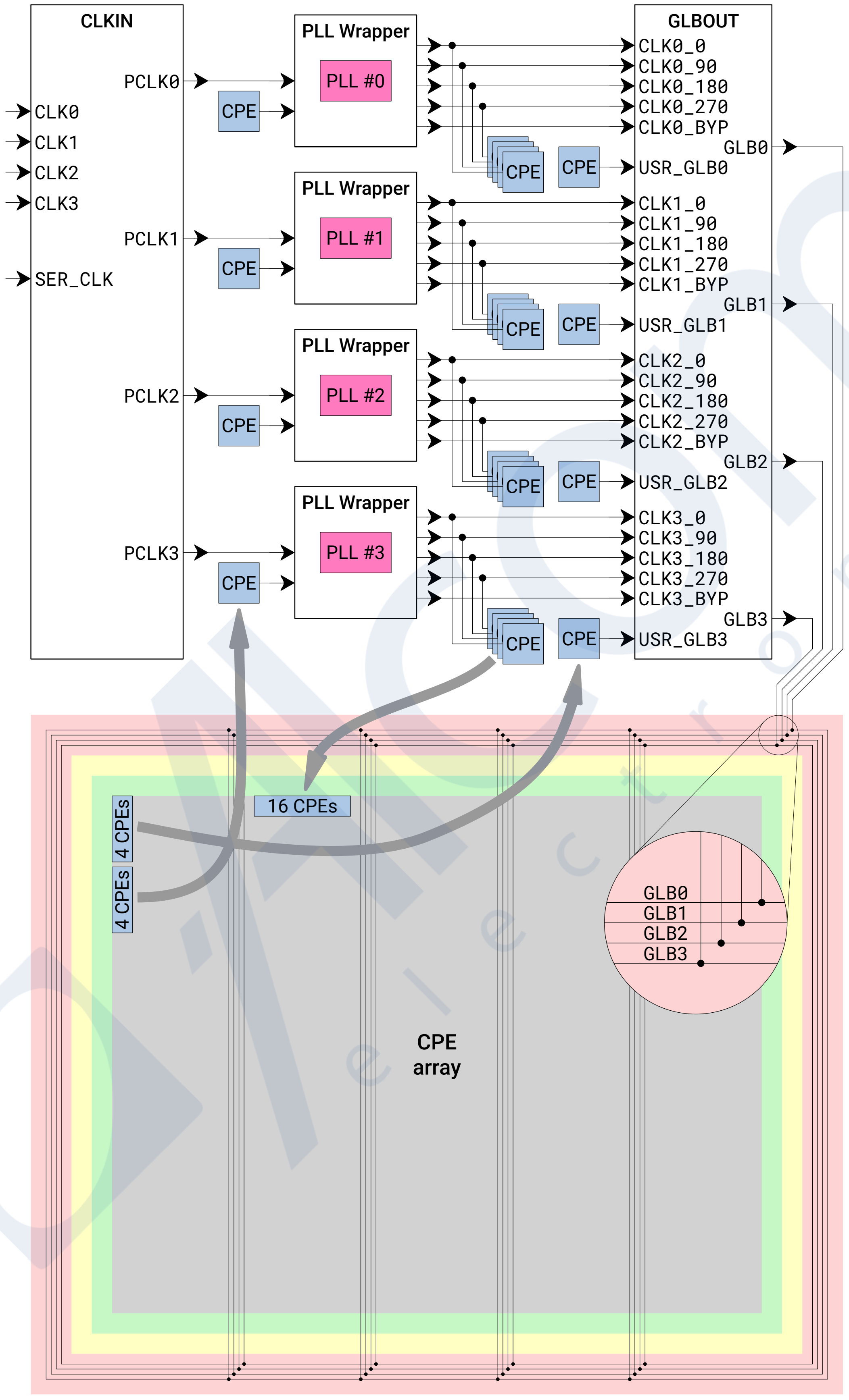


Figure 2.26: *Overview of the Global Mesh signal injection*

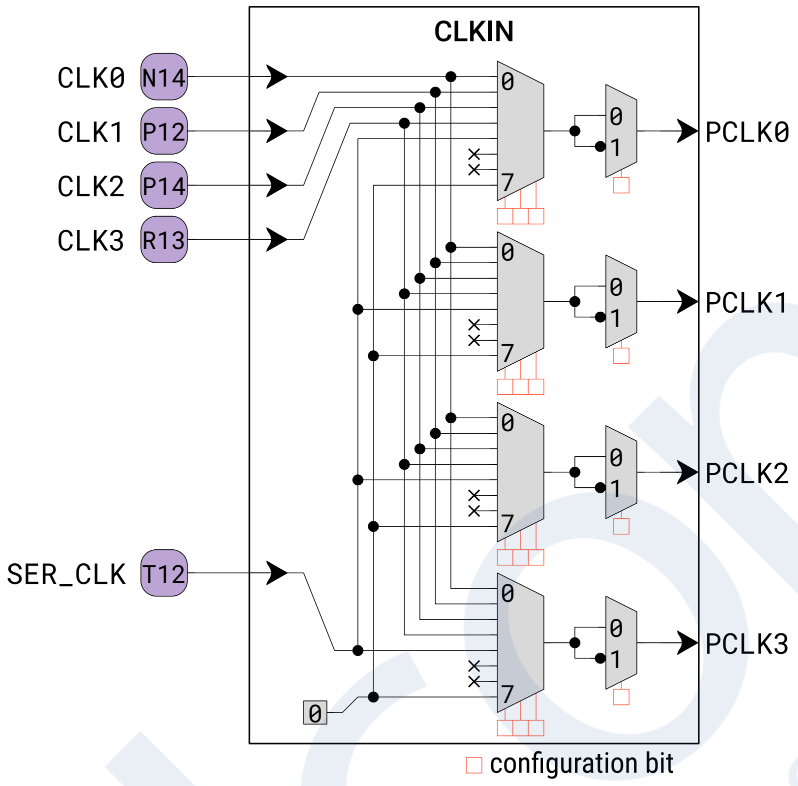


Figure 2.27: *Clock input multiplexers CLKIN*

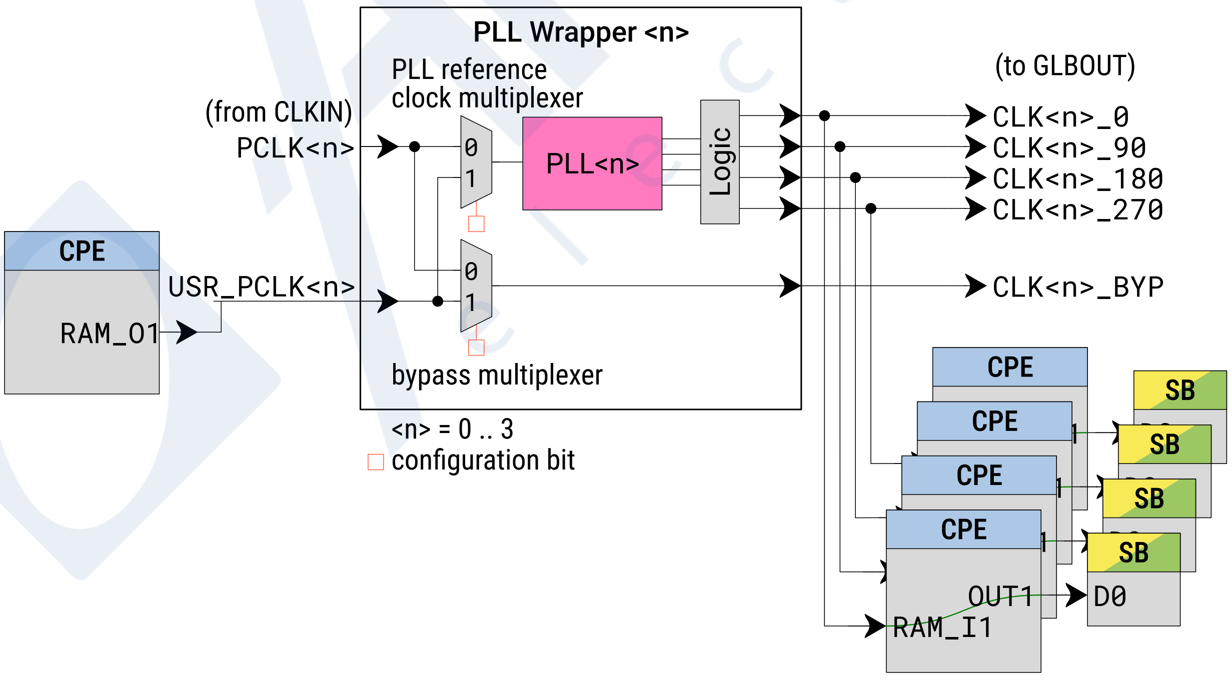


Figure 2.28: *PLL wrapper with input and output clocks*

The four PLL wrapper output clocks are connected to the GLBOUT module (see page 49) 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 2.14.

Table 2.14: *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	CLK0_90	CPE(40,128)	RAM_I1 / OUT1
0	CLK0_180	CPE(41,128)	RAM_I1 / OUT1
0	CLK0_270	CPE(42,128)	RAM_I1 / OUT1
1	CLK1_0	CPE(43,128)	RAM_I1 / OUT1
1	CLK1_90	CPE(44,128)	RAM_I1 / OUT1
1	CLK1_180	CPE(45,128)	RAM_I1 / OUT1
1	CLK1_270	CPE(46,128)	RAM_I1 / OUT1
2	CLK2_0	CPE(47,128)	RAM_I1 / OUT1
2	CLK2_90	CPE(48,128)	RAM_I1 / OUT1
2	CLK2_180	CPE(49,128)	RAM_I1 / OUT1
2	CLK2_270	CPE(50,128)	RAM_I1 / OUT1
3	CLK3_0	CPE(51,128)	RAM_I1 / OUT1
3	CLK3_90	CPE(52,128)	RAM_I1 / OUT1
3	CLK3_180	CPE(53,128)	RAM_I1 / OUT1
3	CLK3_270	CPE(54,128)	RAM_I1 / OUT1

2.6.4 GLBOUT Multiplexers

The GLBOUT multiplexers are connected to five signals from a PLL wrapper (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 2.15. Figure 2.29 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.

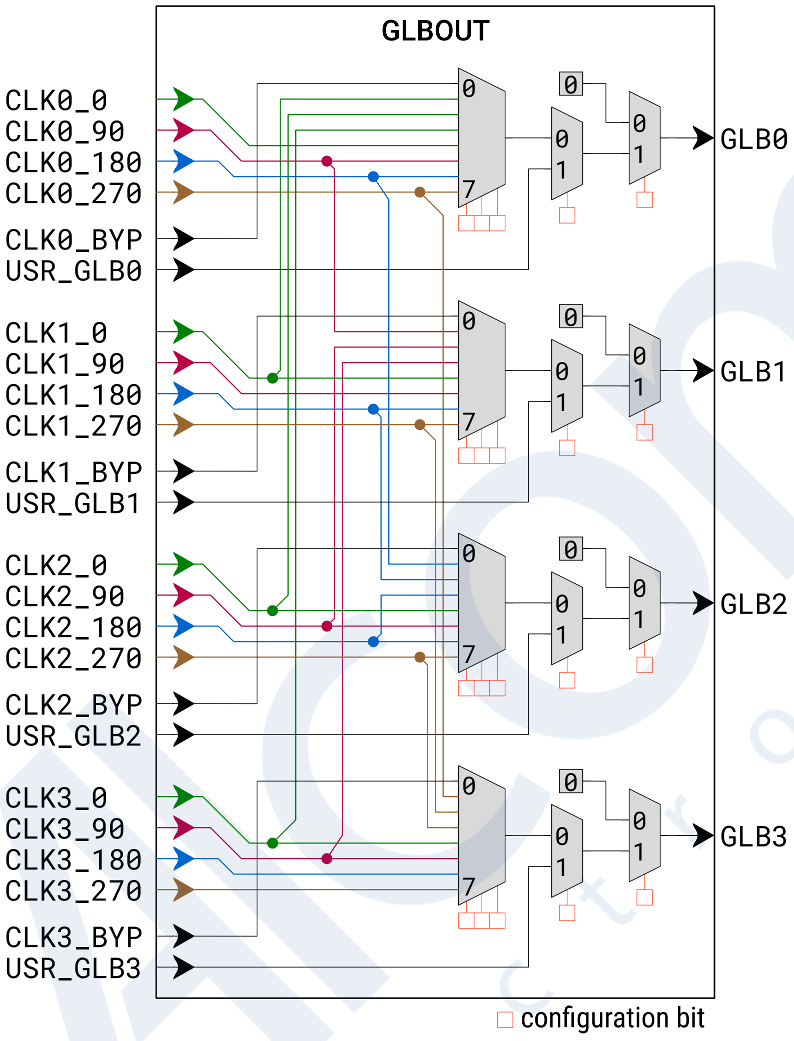


Figure 2.29: *GLBOUT multiplexers*

Table 2.15: *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

2.6.5 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.

2.7 Clocking Schemes

2.7.1 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 2.26 in previous Section 2.6.1). 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 2.7.2.
2. Using the Global Mesh to distribute the Global Mesh signals as shown in Section 2.7.3.
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 2.7.4.

Figure 2.30 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 2.16. The specification `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 2.17 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 2.18, the selection of the clocking scheme is made by the settings regarding `CC_BUFG` usage and the synthesis option '`-noclkbuf`'.

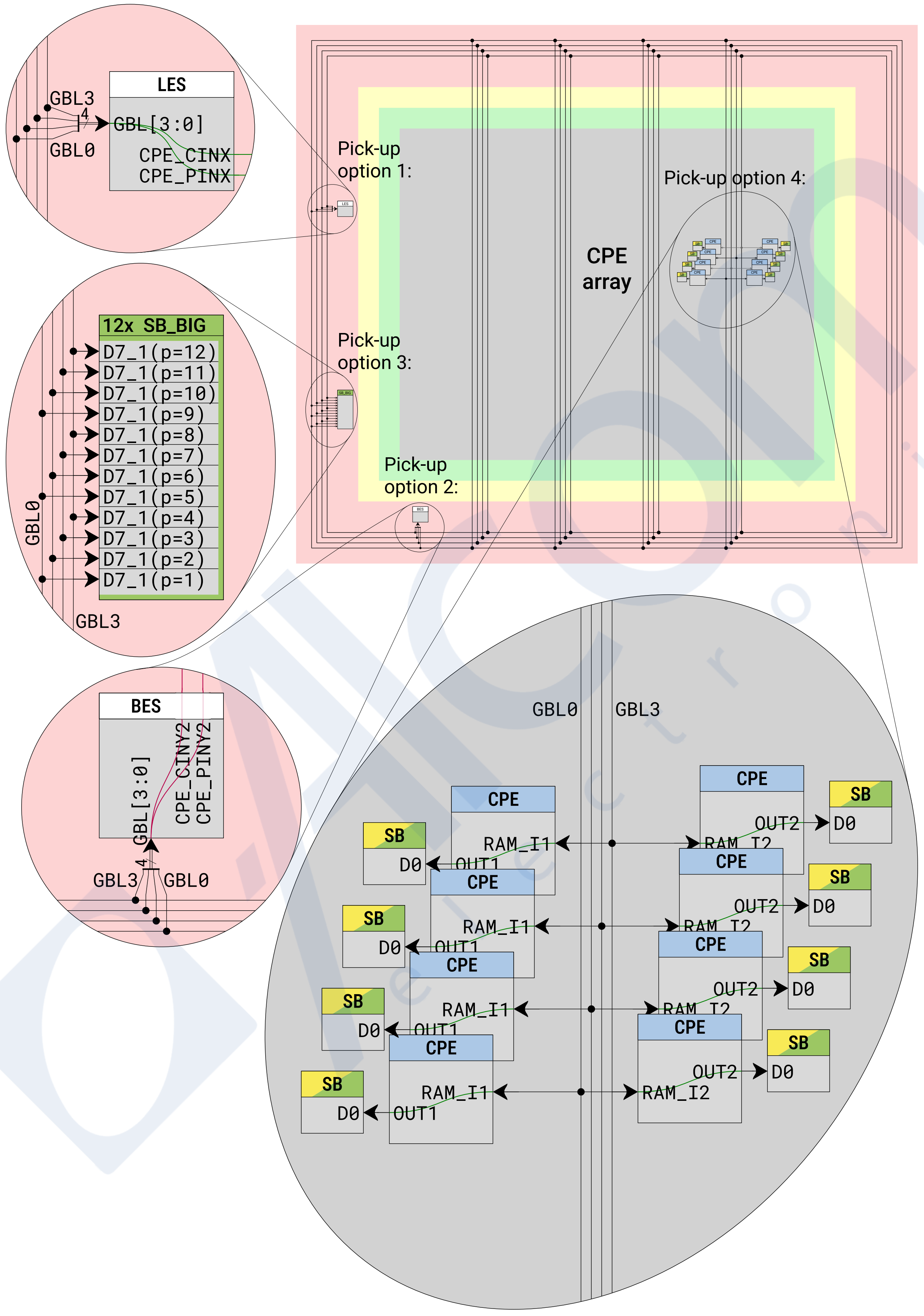


Figure 2.30: *Overview of signal extraction from the Global Mesh*

Table 2.16: *Connecting Global Mesh signals to Switch Boxes (pick-up option 3)*

FPGA edge	Signal	Target	Conditions
Left edge	GLB0 →	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 →	SB_BIG$(x, y, \{2, 6, 10\})$.D7_1	
	GLB2 →	SB_BIG$(x, y, \{3, 7, 11\})$.D7_1	
	GLB3 →	SB_BIG$(x, y, \{4, 8, 12\})$.D7_1	
Bottom edge	GLB0 →	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 →	SB_BIG$(x, y, \{2, 6, 10\})$.D7_2	
	GLB2 →	SB_BIG$(x, y, \{3, 7, 11\})$.D7_2	
	GLB3 →	SB_BIG$(x, y, \{4, 8, 12\})$.D7_2	
Right edge	GLB0 →	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 →	SB_BIG$(x, y, \{2, 6, 10\})$.D7_3	
	GLB2 →	SB_BIG$(x, y, \{3, 7, 11\})$.D7_3	
	GLB3 →	SB_BIG$(x, y, \{4, 8, 12\})$.D7_3	
Top edge	GLB0 →	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 →	SB_BIG$(x, y, \{2, 6, 10\})$.D7_4	
	GLB2 →	SB_BIG$(x, y, \{3, 7, 11\})$.D7_4	
	GLB3 →	SB_BIG$(x, y, \{4, 8, 12\})$.D7_4	

Table 2.17: *Connecting Global Mesh signals to CPEs (pick-up option 4)*

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

Table 2.18: *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	2.7.2	Clock distribution via the routing structure
0	enabled	2.7.4.1	Intake CPE with chained CPE row
		2.7.4.2	Intake BES with chained BES row
		2.7.4.4	Intake LES with chained CPE row
1..4	disabled	2.7.3	Clock distribution via the Global Mesh
1..4	enabled	2.7.4.1	Intake CPE with chained CPE row
		2.7.4.3	Unchained BES row with Global Mesh pick-up
		2.7.4.2	Intake BES with chained BES row
		2.7.4.4	Intake LES with chained CPE row

2.7.2 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 2.30. 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.

2.7.3 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 2.30 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.

2.7.4 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 2.31 to 2.34.

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 2.7.4.1 on page 58)
2. Intake BES with chained BES row (Section 2.7.4.2 on page 59)
3. Unchained BES row with Global Mesh pick-up (Section 2.7.4.3 on page 60)
4. Intake LES with chained CPE row (Section 2.7.4.4 on page 61)

2.7.4.1 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 2.31) 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.

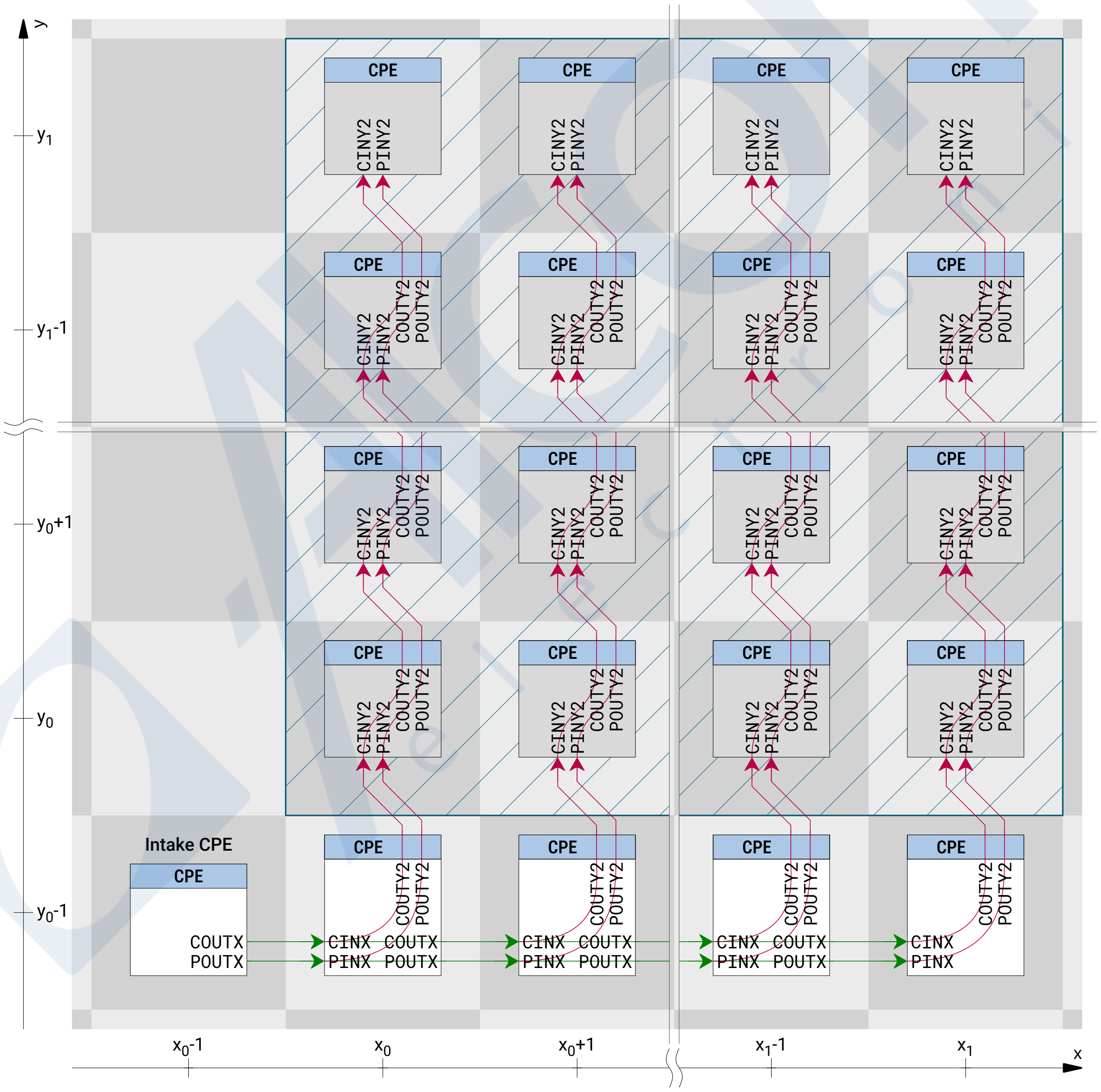


Figure 2.31: *Intake CPE with chained CPE row*

2.7.4.2 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 2.32) 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.

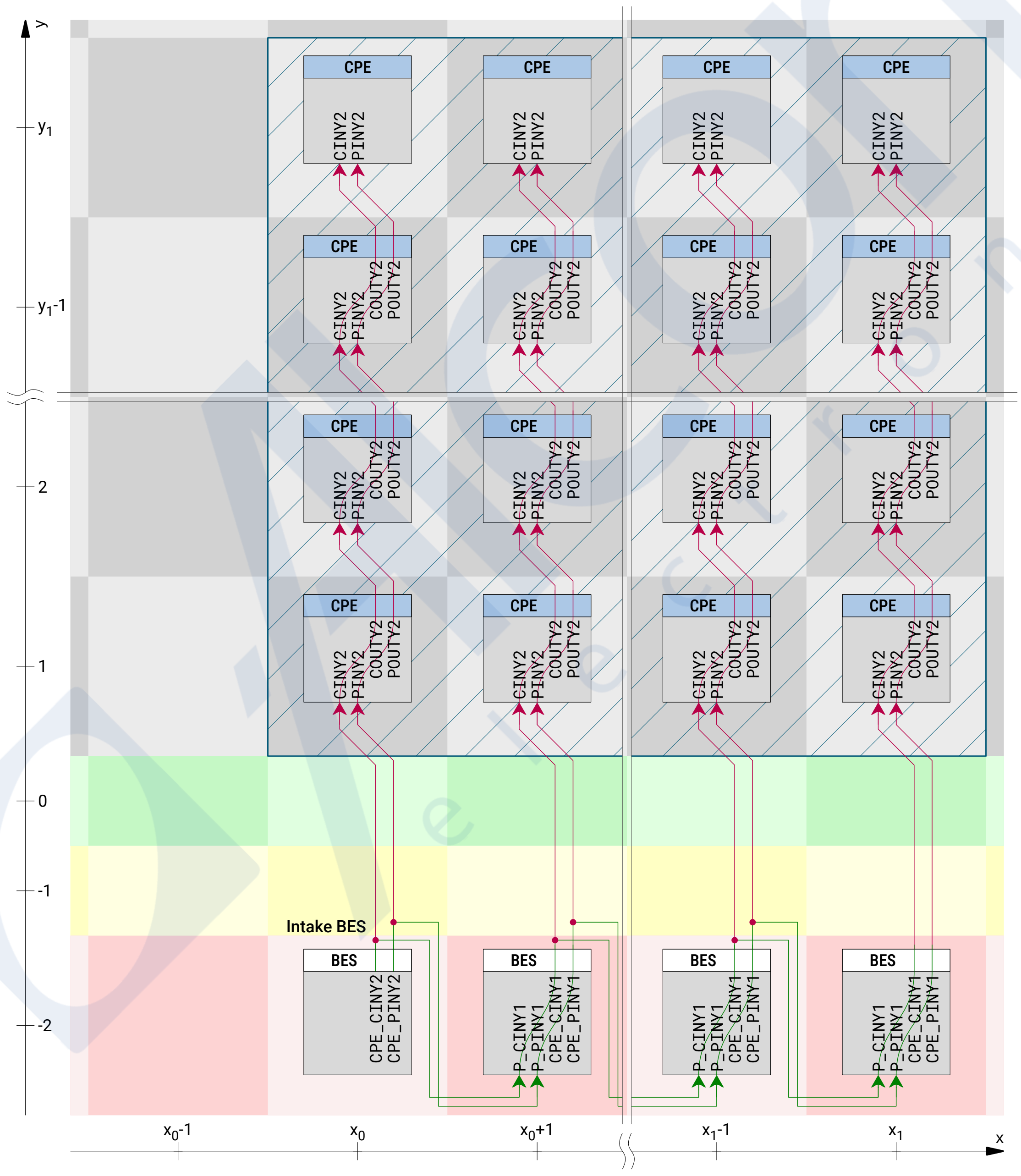


Figure 2.32: *Intake BES with chained BES row*

2.7.4.3 Unchained BES row with Global Mesh pick-up

As a variant of the previous Section 2.7.4.2, 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 2.33. In this case, clock skew occurs only in y direction.

Deviating from Figure 2.33, 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 2.32.

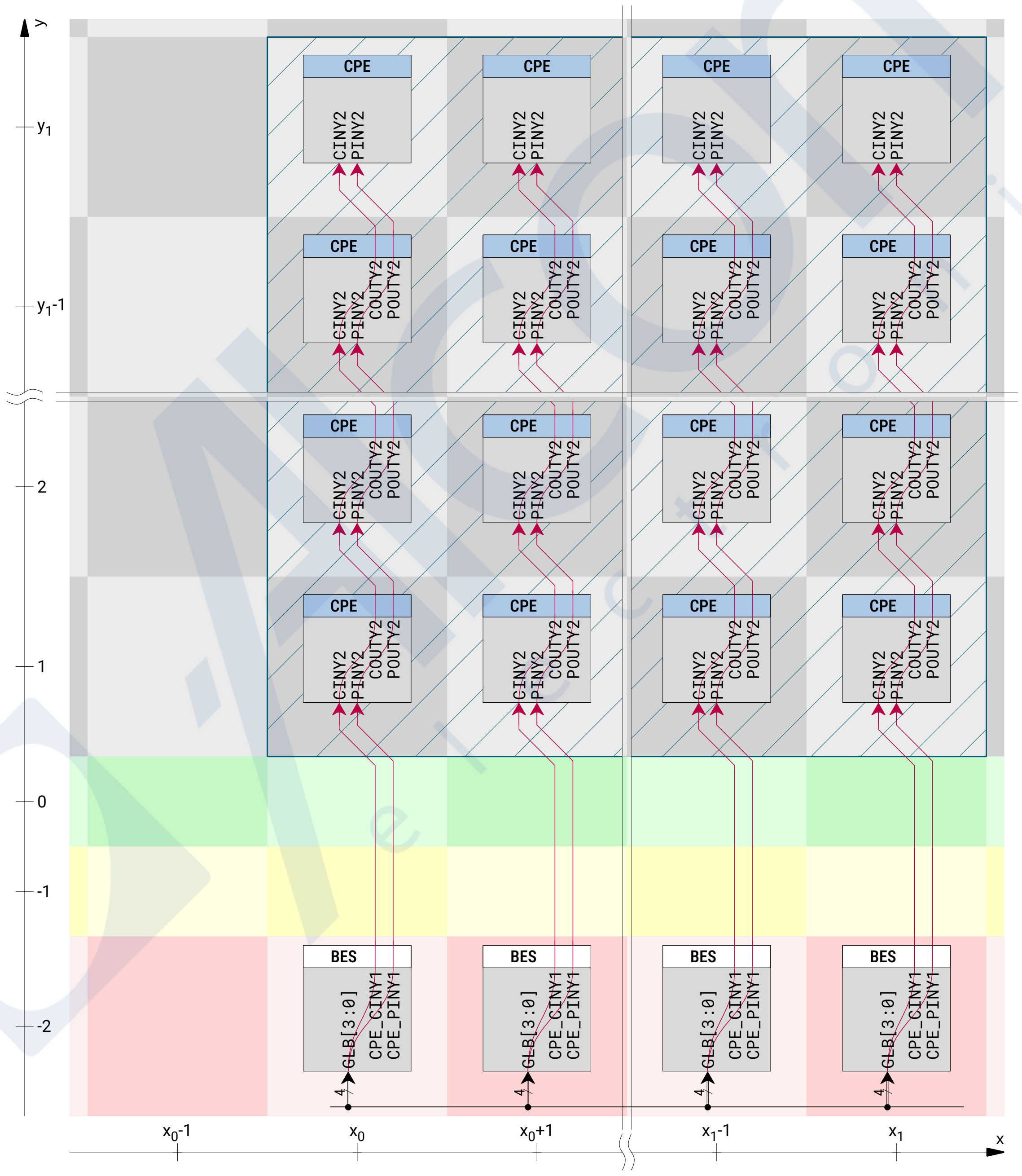


Figure 2.33: *Unchained BES row with Global Mesh pick-up*

2.7.4.4 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 2.34), 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.

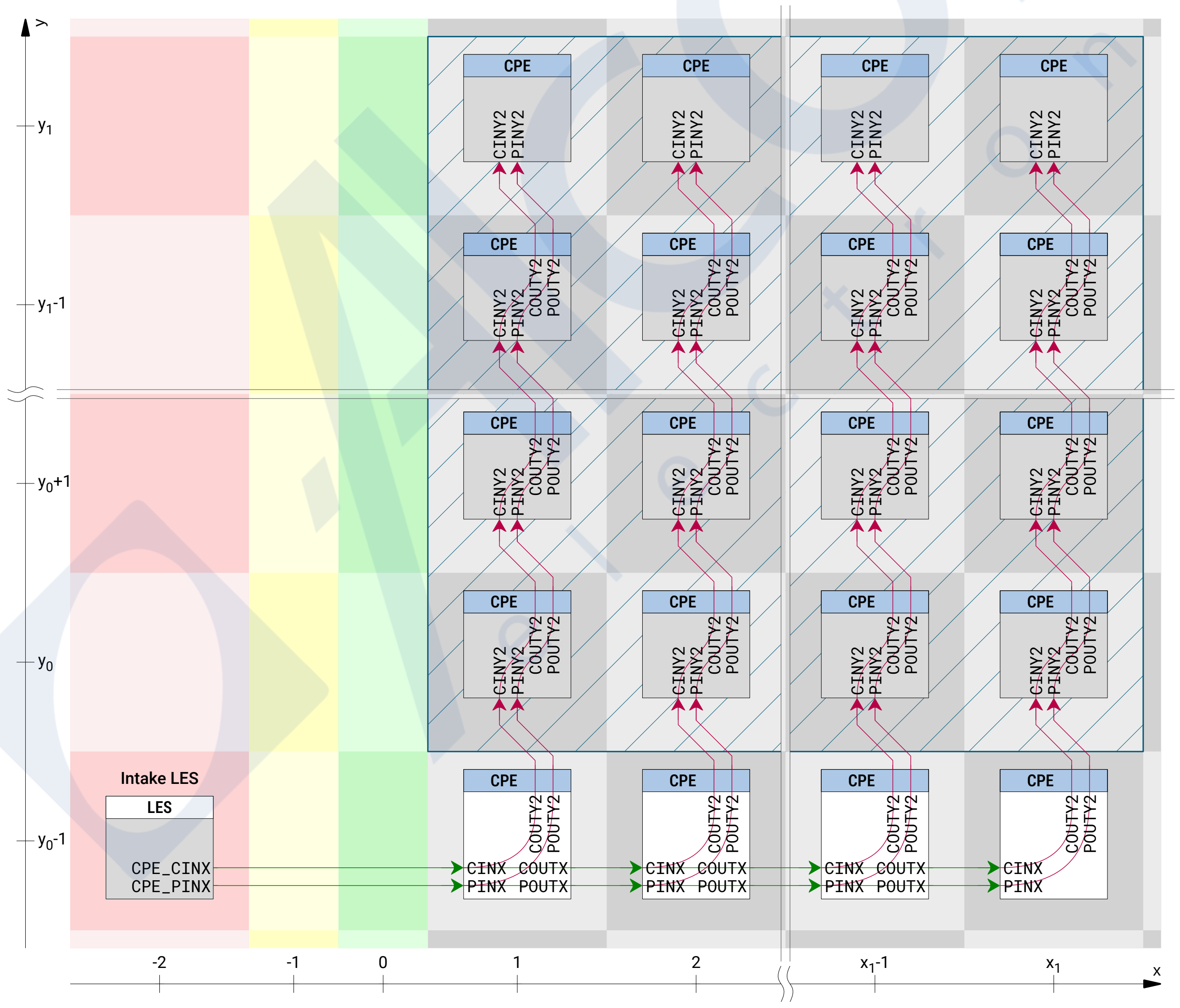


Figure 2.34: *Intake LES with chained CPE row*

2.8 Routing Structure

All functional elements of the GateMate^{TM} 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 $(x, y) = (-1, -1) \dots (162, 130)$ as shown in Figure 2.35.

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

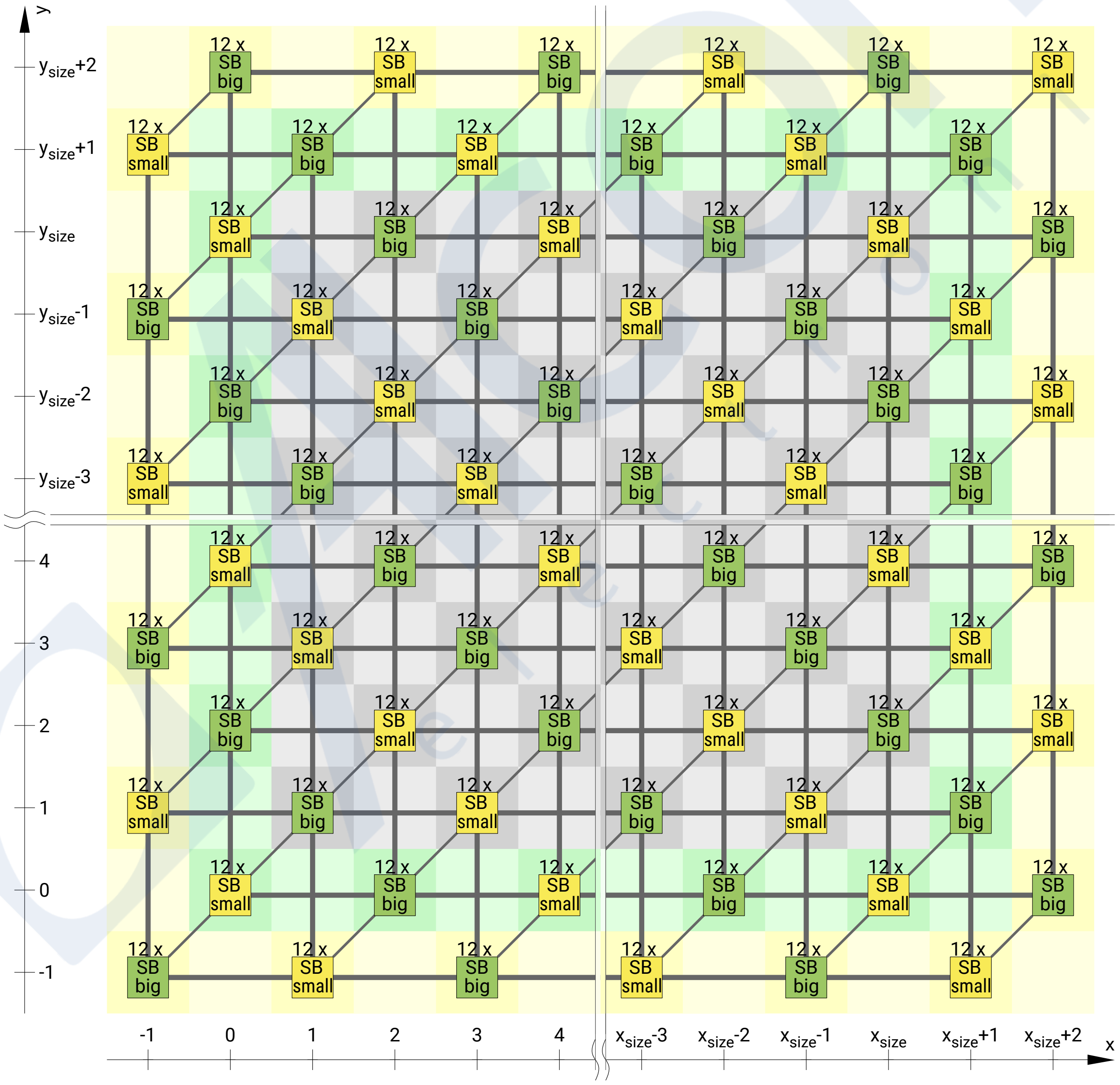


Figure 2.35: *Switch Box (SB) interconnection scheme*

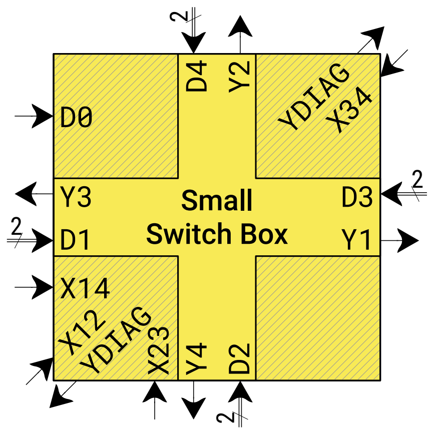


Figure 2.36: *Small Switch Box*

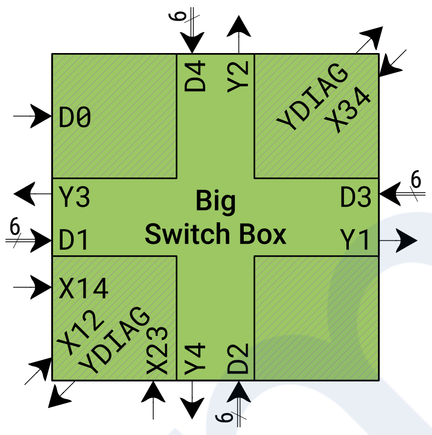


Figure 2.37: *Big Switch Box*

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 Switch Boxes as well.

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

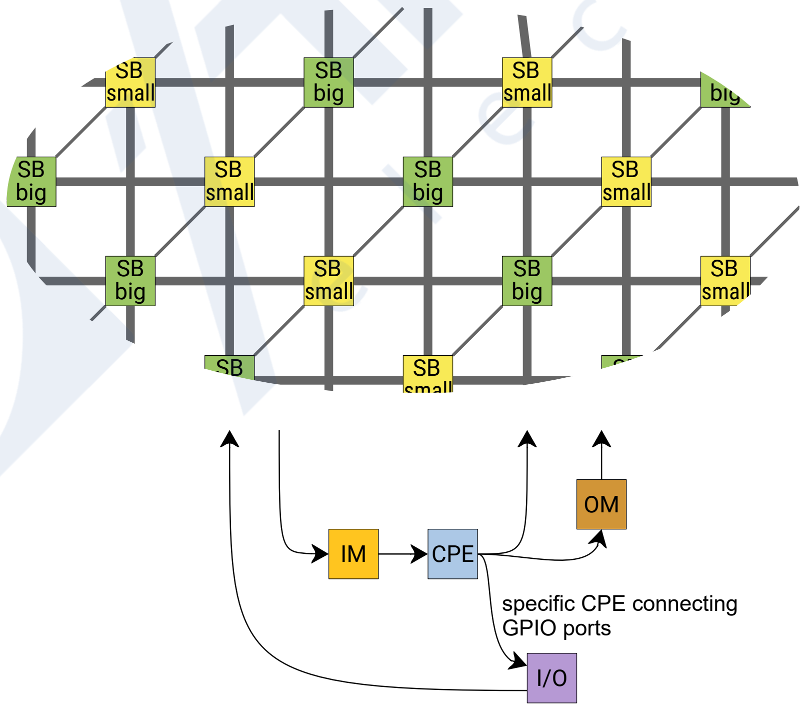


Figure 2.38: *CPE connections to the routing structure*

2.9 Power Supply

The GateMate™ FPGA has 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.

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 application 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 core voltage range. VDD_SER and VDD_SER_PLL have a voltage range from 1.0 V ± 50 mV to 1.1 V ± 50 mV. All 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.

Chapter 3

Workflow and Hardware Setup

3.1 FPGA Workflow

3.1.1 Overview

The workflow from design entry to the GateMate^{TM} 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.

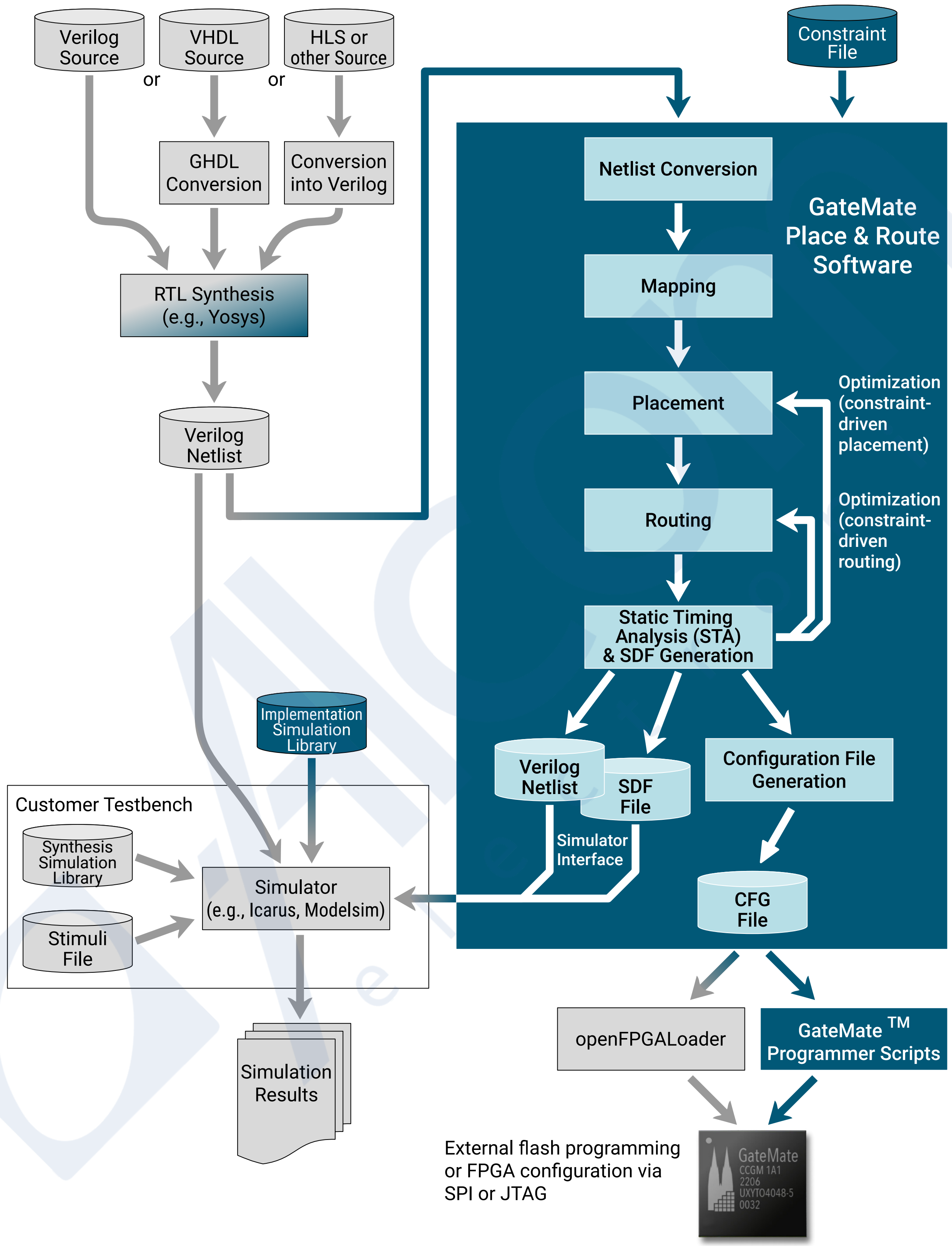

Verilog Source
or
VHDL Source
or
HLS or other Source
GHDL Conversion
Conversion into Verilog
RTL Synthesis (e.g., Yosys)
Verilog Netlist
Constraint File
Netlist Conversion
GateMate Place & Route Software
Mapping
Placement
Optimization (constraint-driven placement)
Routing
Optimization (constraint-driven routing)
Static Timing Analysis (STA) & SDF Generation
Verilog Netlist
SDF File
Configuration File Generation
Simulator Interface
CFG File
Implementation Simulation Library
Customer Testbench
Synthesis Simulation Library
Stimuli File
Simulator (e.g., Icarus, Modelsim)
Simulation Results
openFPGALoader
GateMate TM Programmer Scripts
External flash programming or FPGA configuration via SPI or JTAG
GateMate CCGM 1A1 2206 UXYT04048-5 0032

Figure 3.1: *GateMate^{TM} 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^{TM} 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.

`-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 output signal.

DELAY_OBF=*{0..15}*
Adds an additional delay of n times the nominal 50 ps to the input 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 and access external flash memory with openFPGALoader for the following boards and cables:

- GateMate™ FPGA Evaluation Board
- 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^{TM} FPGA has 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.

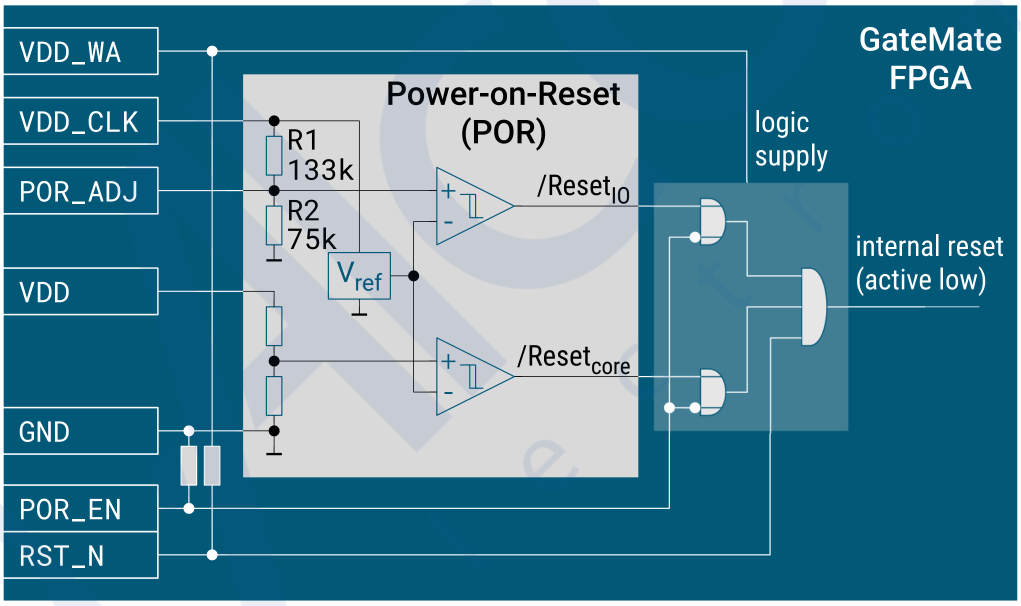


Figure 3.2: *Power-on reset (POR) module block diagram*

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 4.6 on page 98 for exact threshold values in different operating cases.

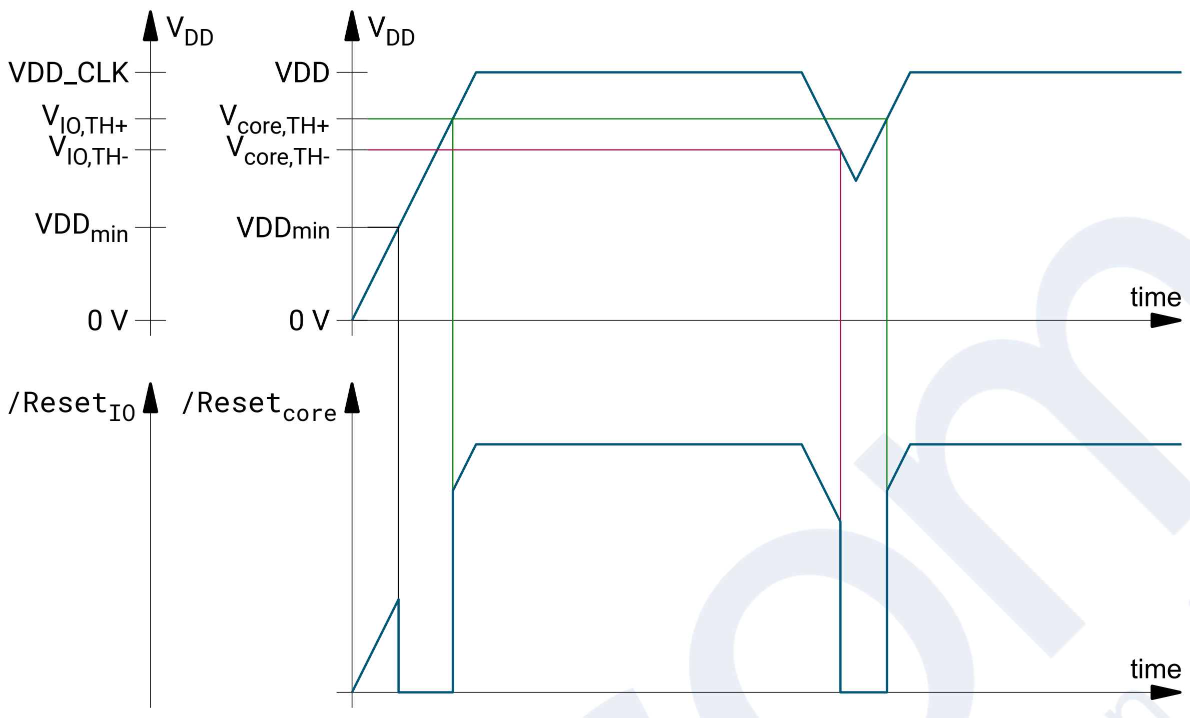


Figure 3.3: *Threshold principles of POR*

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. 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:

- 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 $\leqslant 100\,\mathrm{k\Omega}$.
- 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

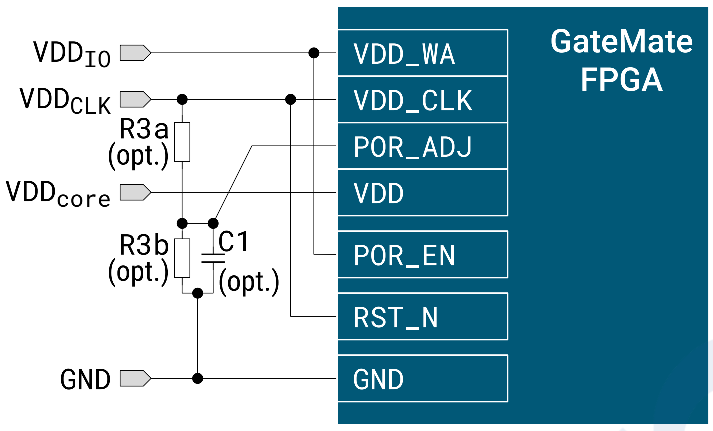


Figure 3.4: *FPGA reset using the POR module*

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.

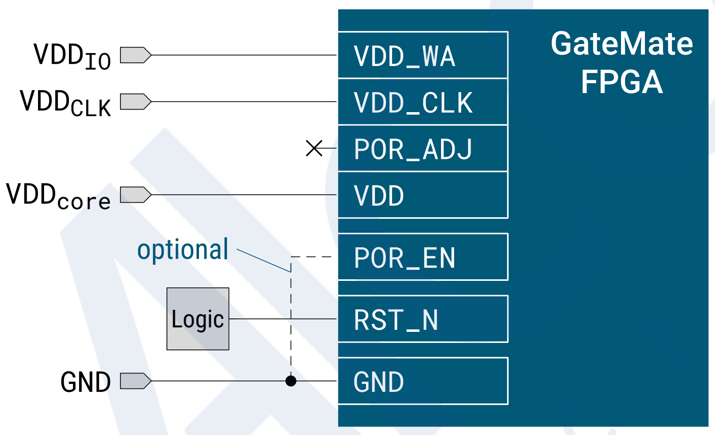


Figure 3.5: *FPGA reset driven by an external logic*

3.3 Configuration Procedure

3.3.1 Configuration Modes

The GateMate^{TM} 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^{TM} 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.

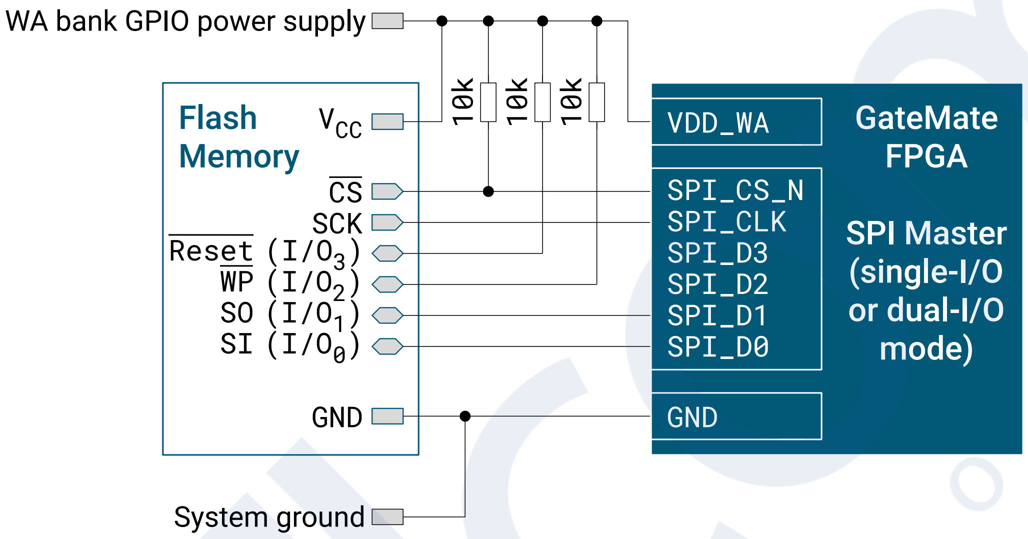


Figure 3.6: *Configuration data from flash memory in SPI single-I/O or dual-I/O mode*

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.

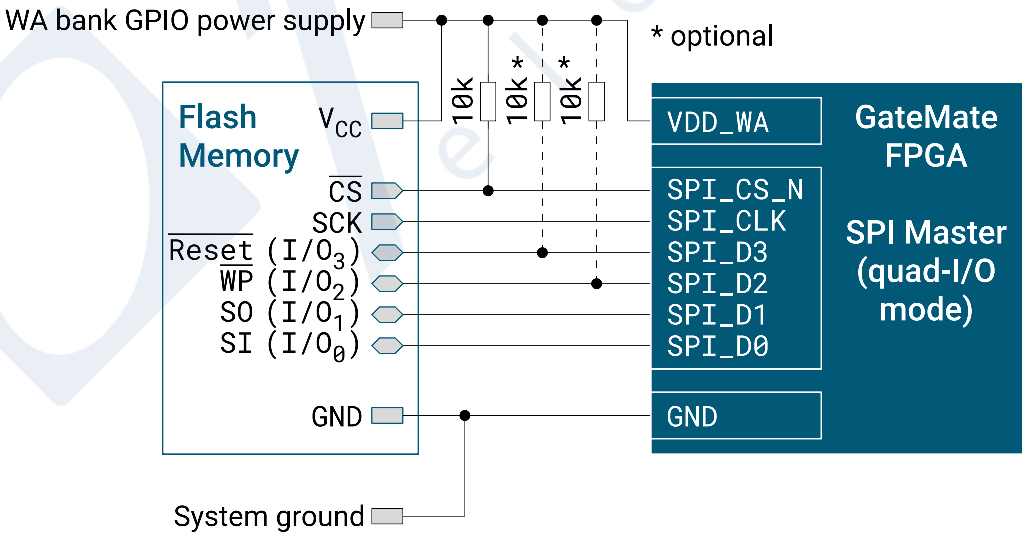


Figure 3.7: *Configuration data from flash memory in SPI quad-I/O mode*

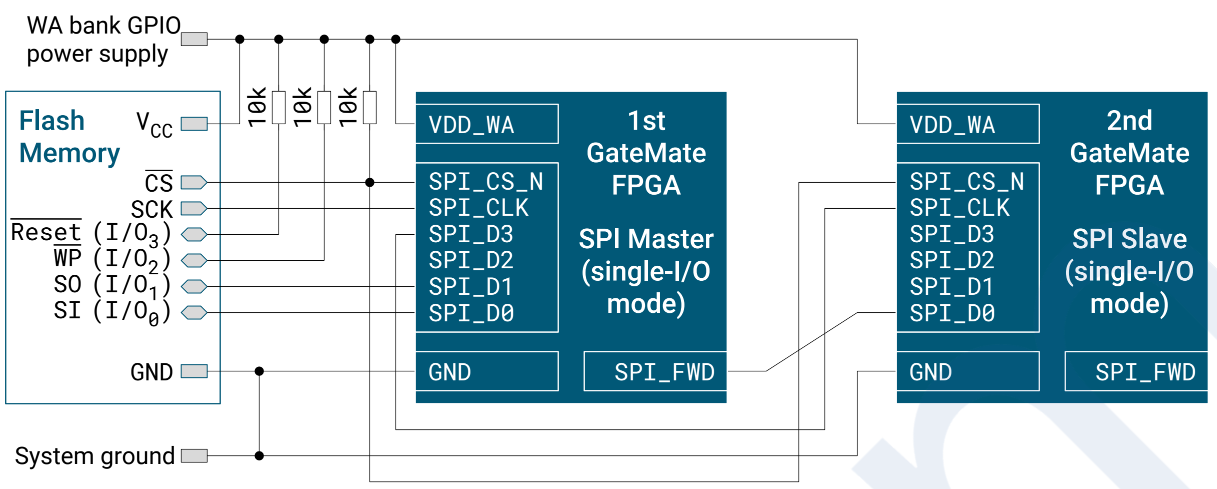


Figure 3.8: *Configuration data stream from a single flash memory*

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. 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.

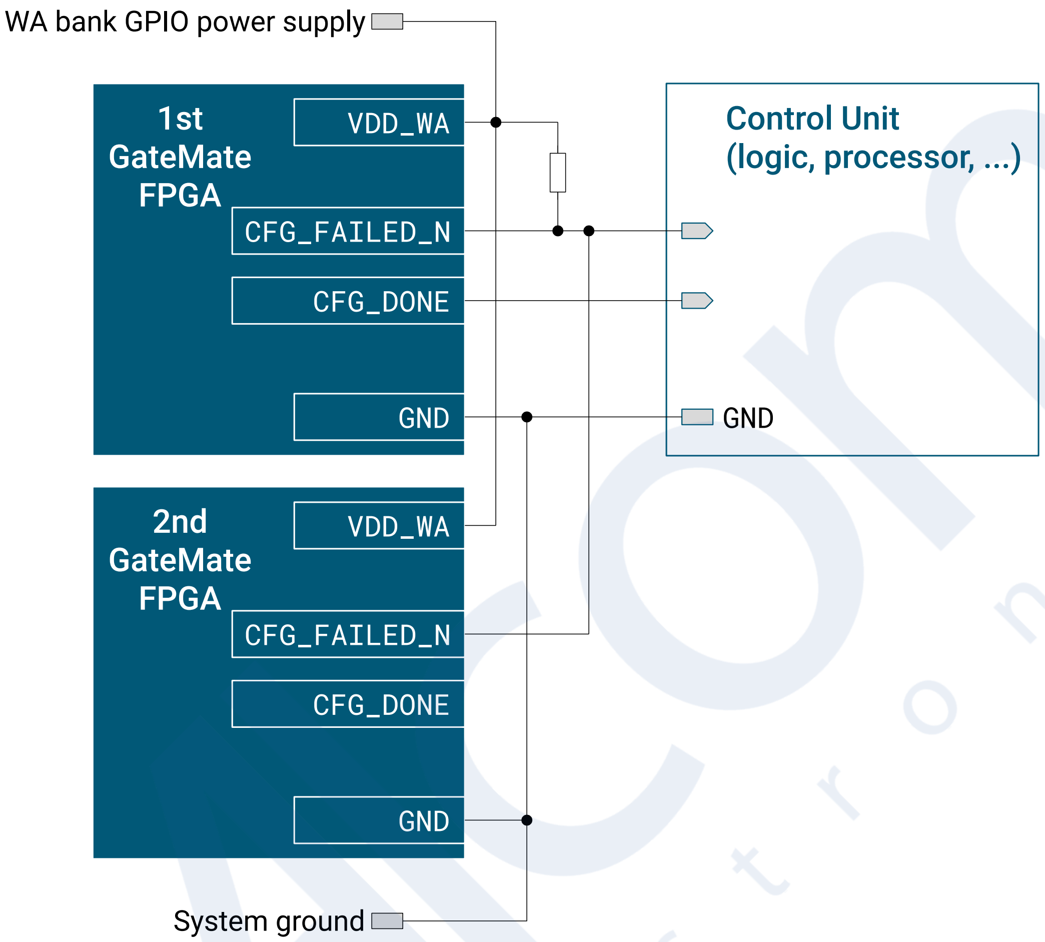


Figure 3.9: *Connection of CFG_DONE and CFG_FAILED_N signals in multi-FPGA applications with a single flash memory*

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 4.3. 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.

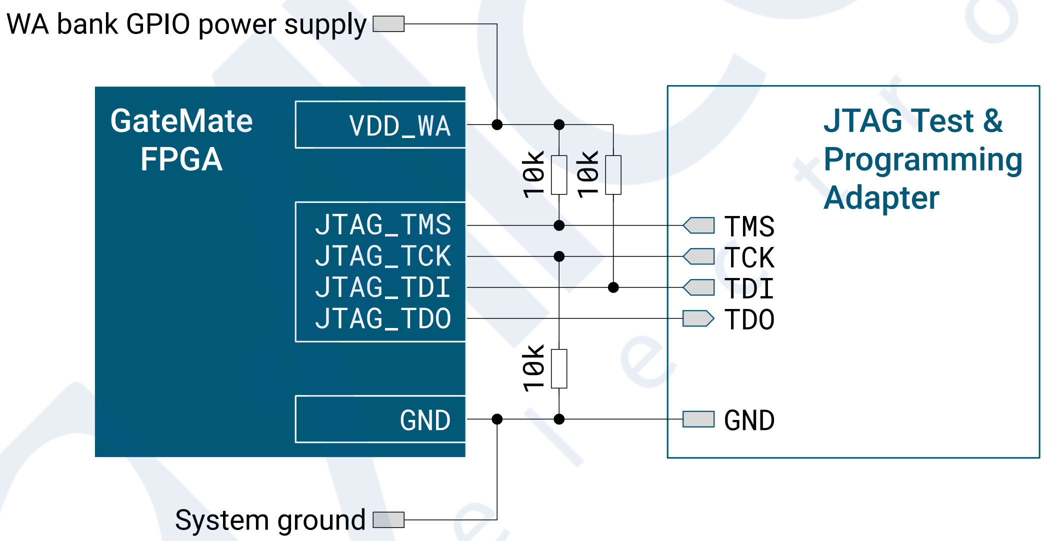


Figure 3.10: *JTAG connection scheme*

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.

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 3.11.

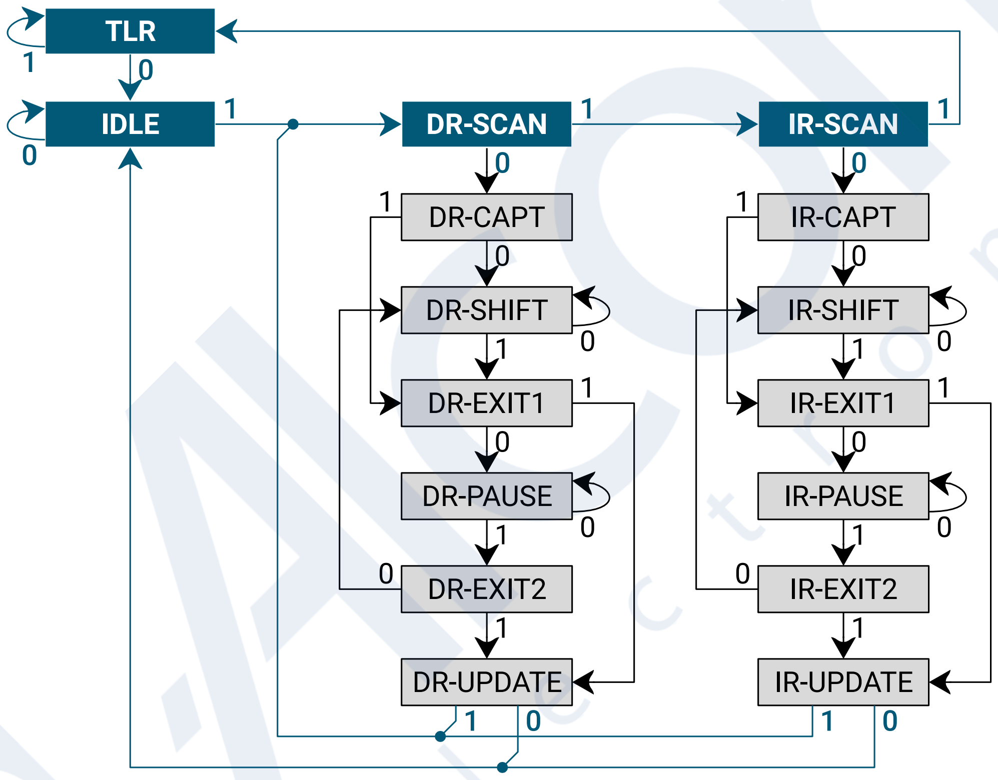


Figure 3.11: *Finite-state machine of the JTAG interface*

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 3.12 and 3.13. 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.

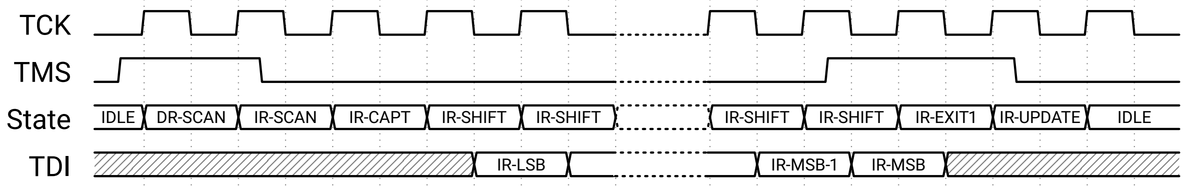


Figure 3.12: *JTAG instruction shift NEW COMMAND*

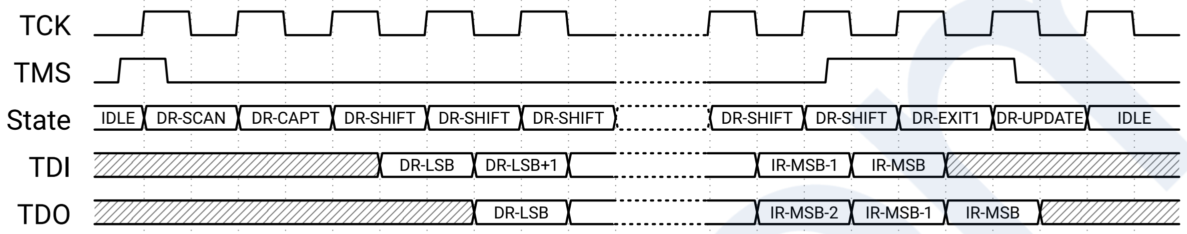


Figure 3.13: *JTAG data shift*

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:	`0x3F`	**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 `0xC`. 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:	0x06	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 3.14 illustrates the wiring between a single FPGA connected to an external SPI component and a JTAG host controller.

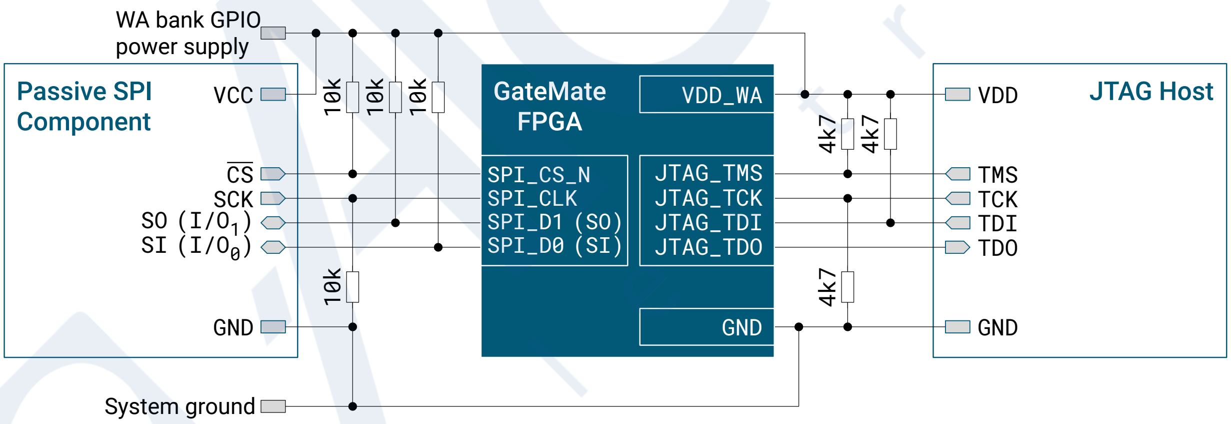


Figure 3.14: *JTAG-SPI bypass connection scheme*

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 `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.

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

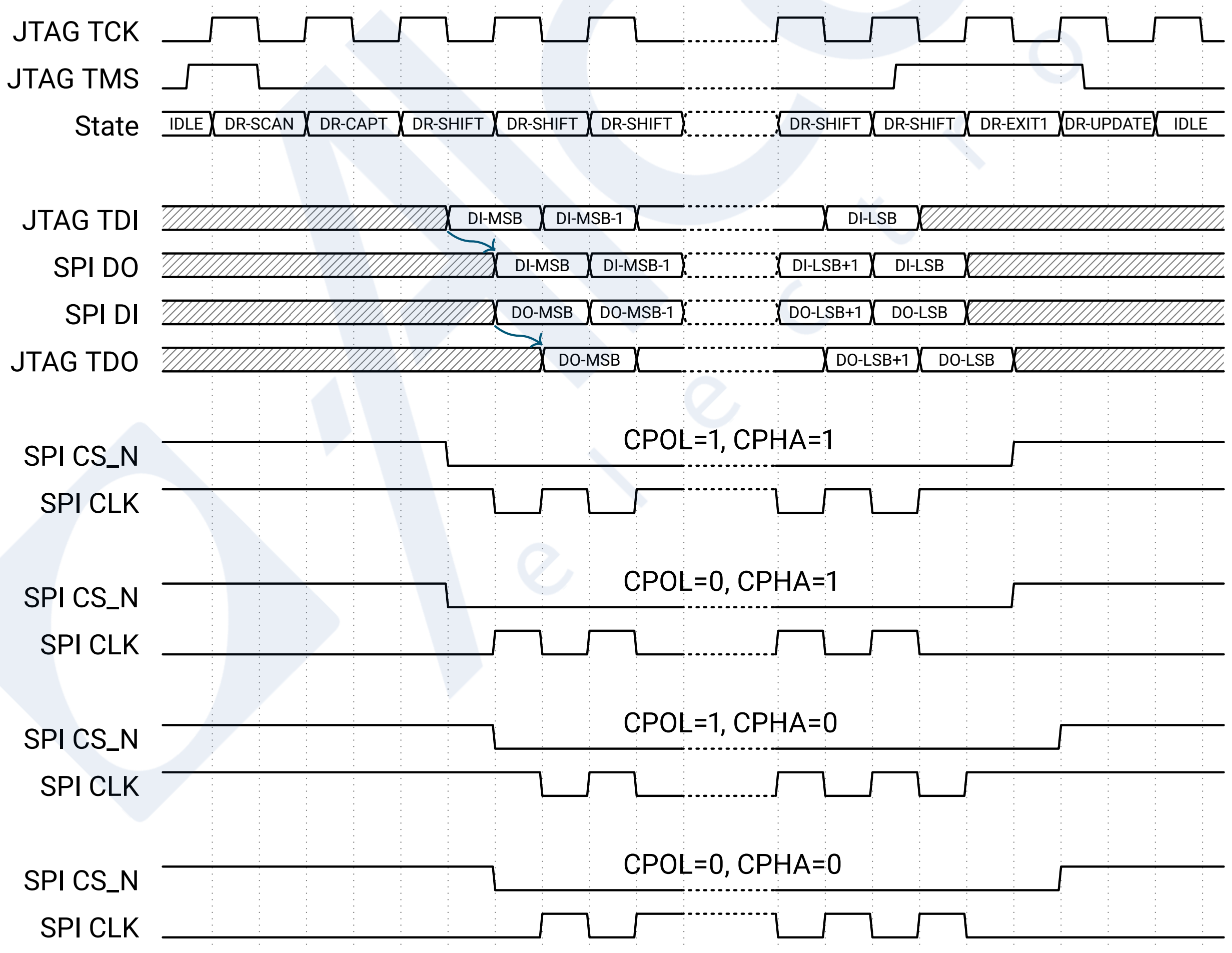


Figure 3.15: *Timing diagram of the JTAG-SPI bypass mode.*

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 LSB first. The actual SPI transfer start 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. 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:	0x 01	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:	0x 02	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 0x 1 – 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 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** Set value in testmode register.	Shift width: 2 bits
Shift-in bits:	[1:0]	Testmode to be set: 0b00: Disable testmode 0b01: reserved 0b10: reserved 0b11: Boundary-Scan Test	
Shift-out bits:	[1:0]	Don't care	

Instruction:	**0x3D**	**JTAG_SAMPLE_PRELOAD** Boundary-scan SAMPLE and PRELOAD instruction. The register 0x20 – Set Testmode must be set to 0b11.	Shift width: 317 bits
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:	**0x3E**	**JTAG_EXTEST** Boundary-scan EXTEST instruction. The register 0x20 – Set Testmode must be set to 0b11.	Shift width: 317 bits
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
	-40		125	°C	Junction temperature
$\mathrm{VDD}_{\mathrm{IO}}$			2.75	V	I/O supply voltage (GPIO banks and `VDD_CLK`)
$\mathrm{VDD}_{\mathrm{core}}$			1.20	V	Core supply voltage (`VDD`, `VDD_PLL`, `VDD_SER` and `VDD_SER_PLL`)
$\mathrm{V}_{\mathrm{ESD}}$			4000	V	HBM ESD class 2

Table 4.2: *Operation range*

Symbol	Min	Typ	Max	Unit	Description
	−20		85	°C	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
					CCGM1A1 leakage current in low power mode
I_{leak,low_power}		2		mA	at −20 °C
		8		mA	at +20 °C
		96		mA	at +85 °C
					CCGM1A1 leakage current in economy mode
$I_{leak,economy}$		3		mA	at −20 °C
		12		mA	at +20 °C
		130		mA	at +85 °C
					CCGM1A1 leakage current in speed mode
$I_{leak,speed}$		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)

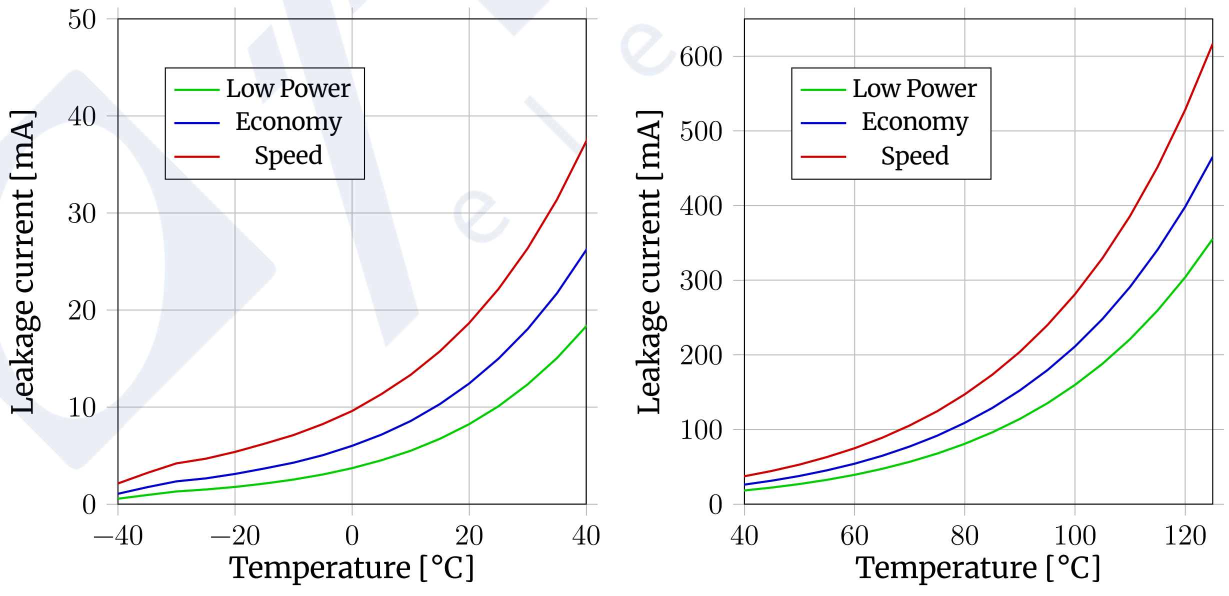


Figure 4.1: *CCGM1A1 leakage current*

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.

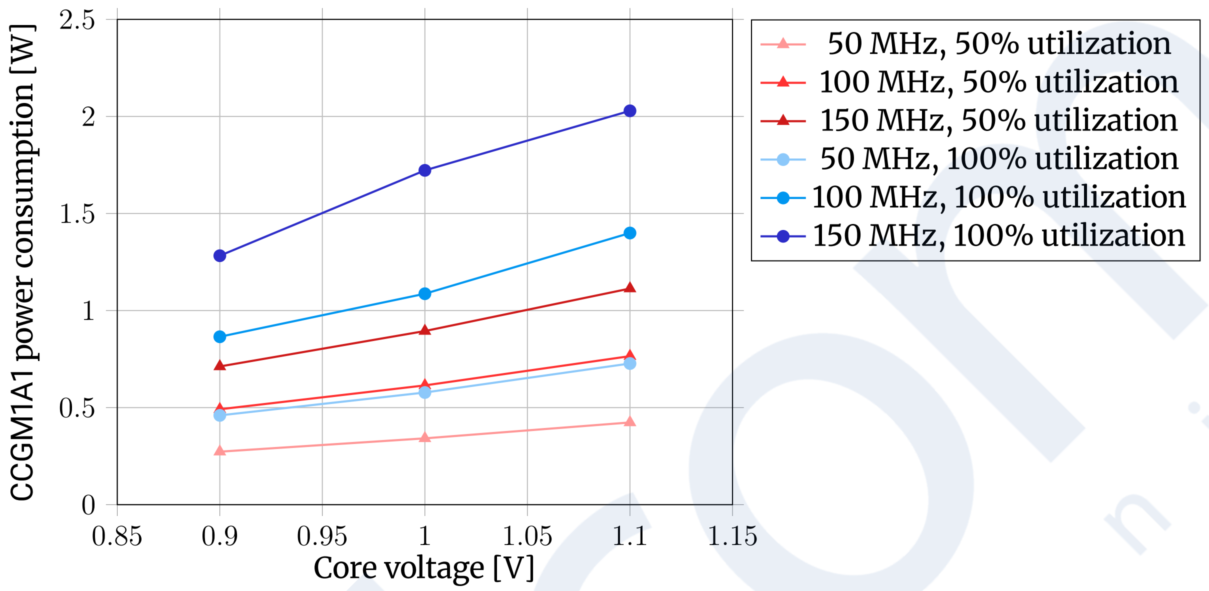


Figure 4.2: *CCGM1A1 total power consumption based on Galois-LFSR*

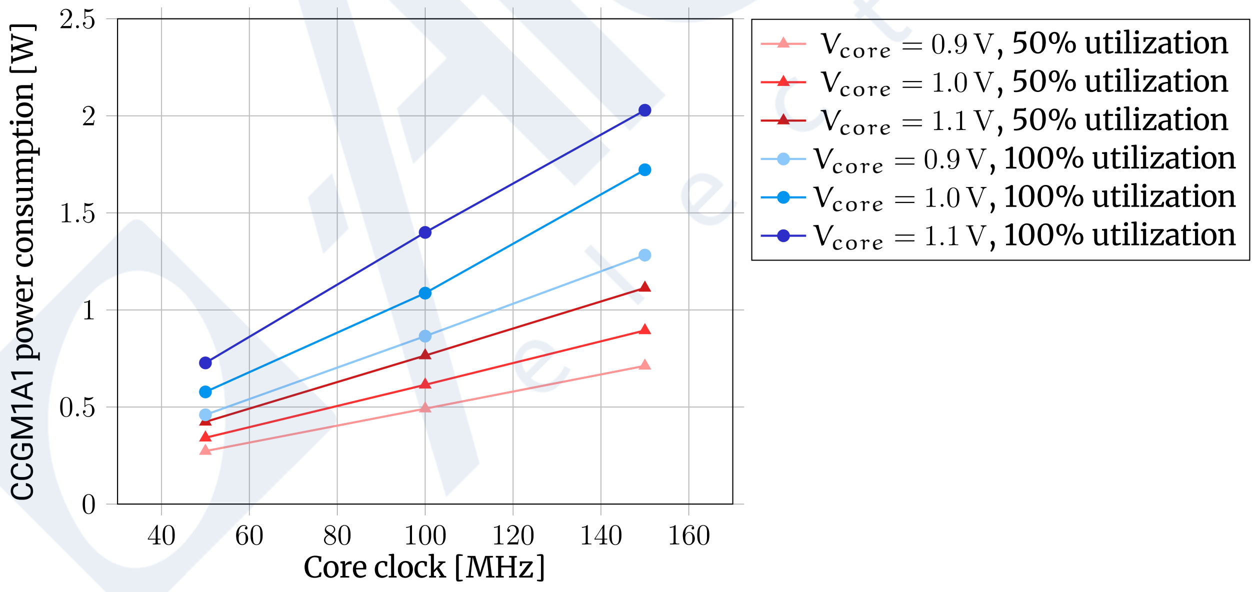


Figure 4.3: *CCGM1A1 total power consumption based on Galois-LFSR*

Table 4.3: *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
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 4.4: *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 4.5: *PLL characteristics*

Symbol	Min	Typ	Max	Unit	Description
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
$f_{PLL,\,low\,power}$			250	MHz	PLL output frequency in low power mode
$f_{PLL,\,economy}$			312.5	MHz	PLL output frequency in economy mode
$f_{PLL,\,speed}$			416.75	MHz	PLL output frequency in speed mode

Table 4.6: *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.
[2] Level adjustment with R3a $= 500\,\mathrm{k\Omega}$ (see Figure 3.4 on page 76).
[3] Level adjustment with R3a $= 250\,\mathrm{k\Omega}$.

Chapter 5

Pinout

5.1 CCGM1A1 324-Ball FBGA Pinout

The GateMate^{TM} FPGA CCGM1A1 has 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.

	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*

N			IO_WA_A8	IO_WA_B8	
P	IO_WA_A7	IO_WA_B7			
R	IO_WA_A6	IO_WA_B6	IO_WA_A5	IO_WA_B5	IO_WA_A0
T		IO_WA_A4	IO_WA_B4		IO_WA_B0
U	IO_WA_A3	IO_WA_B3		IO_WA_A1	
V		IO_WA_A2	IO_WA_B2	IO_WA_B1	
	1	2	3	4	5

Figure 5.2: *Configuration pins of CCGM1A1 FBGA pinout for use as GPIO after configuration*

		CLK0	N
CLK1		CLK2	P
	CLK3		R
12	13	14	

Figure 5.3: *Clock input pins as second function of some GPIOs*

5.2 CCGM1A1 Pinout Description

All ball signals are grouped as shown in Table 5.1. The complete pin list can be found in Table 5.2.

Table 5.1: *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
🟪	serializer / deserializer (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

(continued on next page)

Table 5.1: *Pin types*

(continued from previous page)

Color	Pin type	Number of pins
🟩	Power supply pins for FPGA core and GPIO The CCGM1A1 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 has a single ground level for all supply voltages. All ground pins must be connected.	74
⬜	Any other pins Clock input (single ended or differential), reset input, power-on reset adjustment input and one not connected pin (must be left open) are available in this pin group.	5

Table 5.2: *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

(continued on next page)

Table 5.2: *CCGM1A1 Pin list sorted by ball name*

(continued from previous page)

Ball	Signal name	Signal group	Reset category	Description
⬛ 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
⬛ 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

(continued on next page)

Table 5.2: *CCGM1A1 Pin list sorted by ball name*

(continued from previous page)

Ball	Signal name	Signal group	Reset category	Description
🟧 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
🟧 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

(continued on next page)

Table 5.2: *CCGM1A1 Pin list sorted by ball name*

(continued from previous page)

Ball	Signal name	Signal group	Reset category	Description
🟧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
🟦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

(continued on next page)

Table 5.2: *CCGM1A1 Pin list sorted by ball name*

(continued from previous page)

Ball	Signal name	Signal group	Reset category	Description
🟧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
🟩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

(continued on next page)

Table 5.2: *CCGM1A1 Pin list sorted by ball name*

(continued from previous page)

Ball	Signal name	Signal group	Reset category	Description
🟩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
🟦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

(continued on next page)

Table 5.2: *CCGM1A1 Pin list sorted by ball name*

(continued from previous page)

Ball	Signal name	Signal group	Reset category	Description
🟦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
🟩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

(continued on next page)

Table 5.2: *CCGM1A1 Pin list sorted by ball name*

(continued from previous page)

Ball	Signal name	Signal group	Reset category	Description
🟦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
🟩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

(continued on next page)

Table 5.2: *CCGM1A1 Pin list sorted by ball name*

(continued from previous page)

Ball	Signal name	Signal group	Reset category	Description
🟨 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
🟩 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

(continued on next page)

Table 5.2: *CCGM1A1 Pin list sorted by ball name*

(continued from previous page)

Ball	Signal name	Signal group	Reset category	Description
🟩 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 $\pm$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
🟨 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 $\pm$50 mV)
🟪 U13	SER_TX_P	SerDes	1	Transmit data line, positive LVDS signal
🟦 U14	GND	Ground	0	Ground
🟦 U15	GND	Ground	0	Ground

(continued on next page)

Table 5.2: *CCGM1A1 Pin list sorted by ball name*

(continued from previous page)

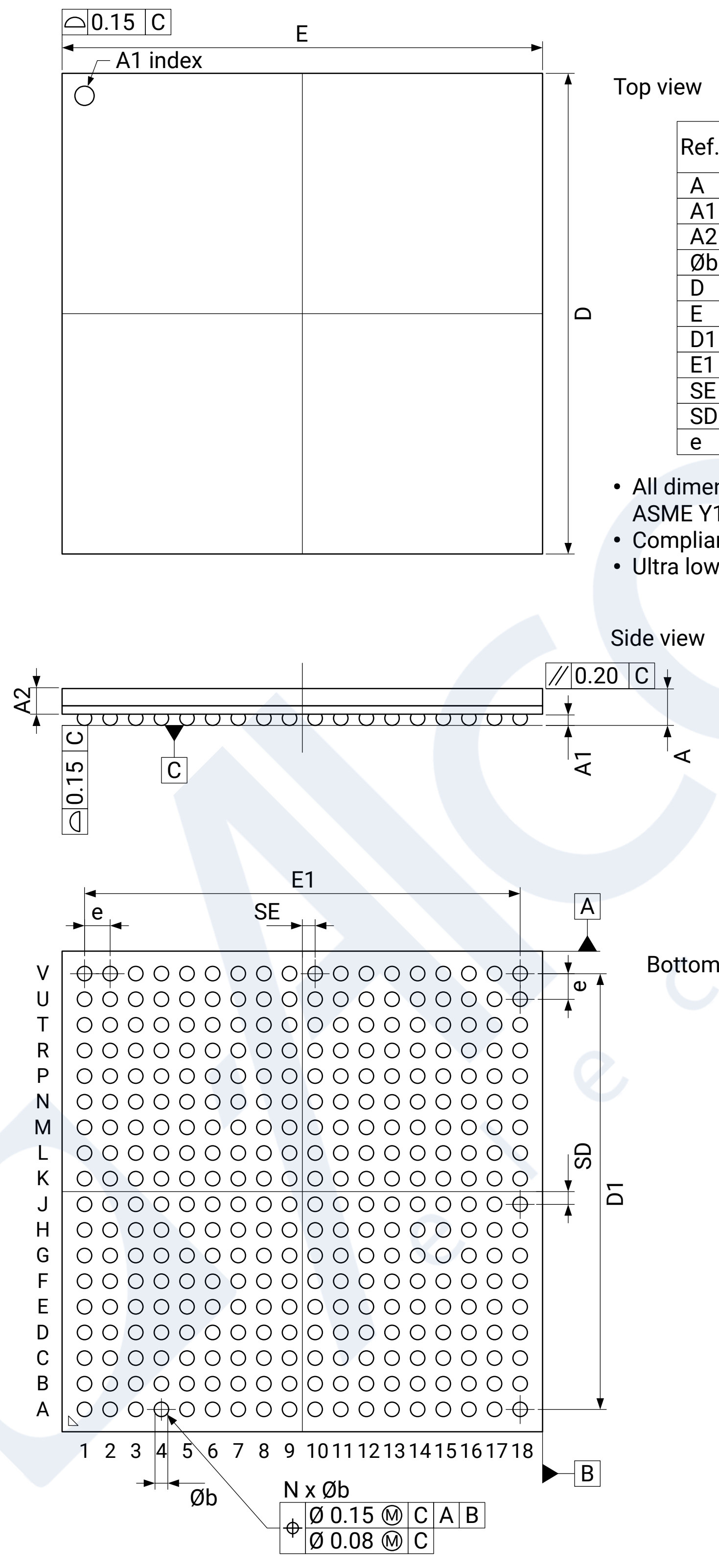
Ball	Signal name	Signal group	Reset category	Description
🟦 U16	`GND`	Ground	0	Ground
🟩 U17	`IO_SB_A0`	GPIO	1	2nd GPIO south bank signal A0
🟩 U18	`IO_SB_B0`	GPIO	1	2nd GPIO south bank signal B0
🟦 V1	`GND`	Ground	0	Ground
🟨 V2	`CFG_FAILED_N`	GPIO	2	1^{st} function: Configuration failed signal
🟨 V2	`IO_WA_A2`	GPIO	1	2^{nd} function: 1st GPIO west bank signal A2
🟨 V3	`CFG_DONE`	GPIO	2	1^{st} function: Configuration done signal
🟨 V3	`IO_WA_B2`	GPIO	1	2^{nd} function: 1st GPIO west bank signal B2
🟨 V4	`CFG_MD3`	GPIO	4	1^{st} function: Configuration mode bit 3
🟨 V4	`IO_WA_B1`	GPIO	1	2^{nd} 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
🟩 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 $\pm$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



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		

- 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 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 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

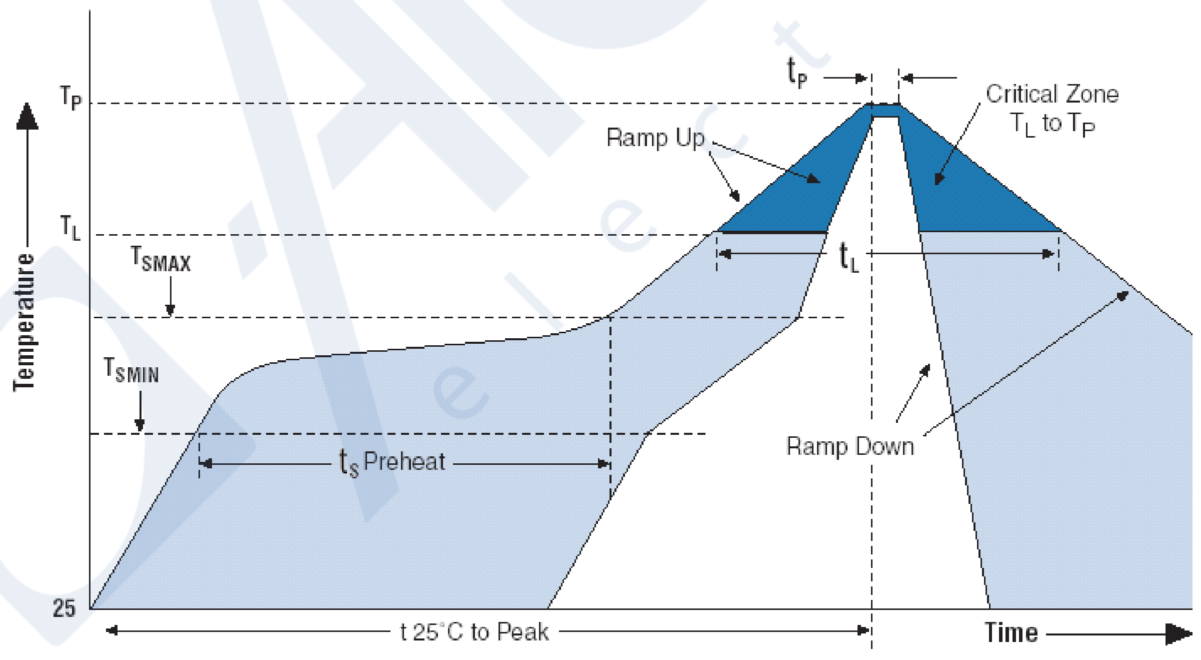


Figure 7.1: *Reflow soldering profile*

Acronyms

ADPLL	all-digital phase-locked loop	45
AI	artificial intelligence	17
BES	Bottom Edge Select	8, 52, 56, 57, 59, 60
BSDL	Boundary Scan Description Language	90
CP-line	carry & propagation signal line	11, 23, 52, 56–58, 69
CPE	Cologne Programmable Element	7, 8, 11, 15, 21–23, 27–29, 33, 46, 49, 52, 56–59, 61, 63, 69, 70
DCO	digitally controlled oscillator	45, 97
DDR	double data rate	18
DPSRAM	dual port SRAM	7, 15, 21, 22, 27, 30, 32, 33, 37
ECC	error checking and correcting	18, 28–30, 34, 35
EMI	electromagnetic interference	56
ESD	electrocstatic discharge	93
FBGA	fine pitch Ball Grid Array	9, 19, 99, 100, 113
FIFO	first-in, first-out memory	7, 8, 11, 18, 28–30, 37–44
FPGA	field-programmable gate array	17, 24, 67, 101

FSM	finite-state machine	8, 84, 87
Galois-LFSR	Galois linear feedback shift register	9, 95
GLB signal	Global Mesh signals GLB[3:0]	49, 51, 52
GPIO	general purpose input / output	7, 9, 12, 15, 18, 21, 24, 25, 45, 46, 63, 64, 70, 71, 93, 95, 99–101
HBM	Human Body Model	93
HDL	Hardware Description Language	51, 65
IM	Input Multiplexer	15, 62, 63
IoT	Internet of Things	17
IPCEI	Important Project of Common European Interest	17
JSON	JavaScript Object Notation (data interchange format)	68
JTAG	Joint Test Action Group	8, 18, 21, 67, 72, 78, 83–90
LES	Left Edge Select	8, 52, 56, 57, 61
LSB	least significant bit	84, 87
LUT	lookup table	67, 68, 95
LUT-2	2-input lookup table	22
LUT-tree	tree-structure lookup table	18, 67
LVCMOS	low voltage CMOS	25
LVDS	low-voltage differential signaling	7, 12, 18, 24, 25, 71, 95
MUX-4	4-input multiplexer	22
OM	Output Multiplexer	15, 62, 63
PCB	printed circuit board	17, 19, 99
PLL	phase-locked loop	8, 11, 12, 15, 18, 21, 45, 46, 49, 64, 81, 94, 97, 101
POR	power-on reset	8, 74–76

RAM	random-access memory	7, 14, 27–31, 33–37, 67
ROM	read-only memory	28, 36
RTL	Register Transfer Level	65
SB	Switch Box	8, 15, 21, 29, 52, 62, 63, 70
SB_BIG	Big Switch Box	52, 56
SDF	Standard Delay Format	65, 69
SDP	simple dual port	7, 11, 18, 27, 28, 30–33, 35, 36, 38, 39
SerDes	serializer / deserializer	15, 18, 21, 45, 64, 94, 101
SLP	Super Low Power	17
SPI	Serial Peripheral Interface	8, 18, 19, 21, 67, 70, 72, 73, 78, 80, 81, 83, 86–90
SRAM	static random-access memory	18, 22, 33, 38
STA	static timing analysis	65
TAP	Test Access Point	83–88
TCK	Test Clock	83, 84
TDI	Test Data Input	83, 84
TDO	Test Data Output	83, 84
TDP	true dual port	7, 11, 18, 27, 28, 30–33, 35, 36, 38, 39
TMS	Test Mode Select	83, 84
VHDL	Very High Speed Integrated Circuit Hardware Description Language	65
Yosys	Yosys Open SYnthesis Suite	14, 65, 67, 68