Arm® Mali™ - IV009 Configuration and Integration Manual

Version r0p0-00rel0 / Revision r0p0

Revision Information

Date	Issue	Confidentiality	Change
16/03/2018	EAC	Confidential	Initial Release
14/09/2018	REL	Confidential	Initial Release

Confidentiality Status

This document is Confidential. This document may only be used and distributed in accordance with the terms of the agreement entered into by Arm and the party that Arm delivered this document to.

Product Status

The information in this document is Final, that is for a developed product.

Contents

List of Tables

Preface

This preface introduces the Arm® Mali™ IV009 Configuration and Integration Manual. It contains the following sections:

- About this book.
- Feedback.

About this book

This book is for the Arm Mali-IV009.

Implementation obligations

This book is designed to help you implement an Arm product. The extent to which the deliverables may be modified or disclosed is governed by the contract between ARM and Licensee. There may be validation requirements, which if applicable will be detailed in the contract between Arm and Licensee and which if present must be complied with prior to the distribution of any silicon devices incorporating the technology described in this document. Reproduction of this document is only permitted in accordance with the licences granted to Licensee.

Arm assumes no liability for your overall system design and performance, the verification procedures defined by Arm are only intended to verify the correct implementation of the technology licensed by Arm, and are not intended to test the functionality or performance of the overall system. You or the Licensee will be responsible for performing any system level tests.

You are responsible for any applications which are used in conjunction with the Arm technology described in this document, and to minimize risks adequate design and operating safeguards should be provided for by you. Arm's publication of any information in this document of information regarding any third party's products or services is not an express or implied approval or endorsement of the use thereof.

Product revision status

The rmpn identifier indicates the revision status of the product described in this book, for example, r1p2, where:

rm Identifies the major revision of the product, for example, r1.

pn Identifies the minor revision or modification status of the product, for example, p2.

Intended audience

This book is for system designers, system integrators, and verification engineers who are designing a System-on-Chip (SoC) device that uses the Mali-IV009 Image Signal Processor.

Using this book

This book is organized into the following sections:

Section	Description
Introduction	This section describes the Mali-IV009 block diagram.
Development tools	This section lists the development tools necessary for the configuration and integration activities.
The implementation process	This section describes the implementation process in details and provides information about the various controls, constraints, inputs, outputs, and resources associated with the implementation process.
Unpacking the deliverables	This section provides information on how to unpack the deliverables and the contents of the directory structure.
Release package sanity check	This section provides information about how to compile the RTL and run the RAM integration testbench.
Configuring the Mali-IV009	This section describes the various configuration parameters and the configuration process.
Clock gate cell integration	This section provides information on how to replace the provided manual clock gating cells.
Memory integration	This section describes the Mali-IV009 memory organization and how to integrate memory blocks into the display processor.
RAM integration test bench	This section provides information related to running the RAM integration test bench.
Mali-IV009 top interfaces	This section describes the core clock, video, memory and other interfaces available with the Mali-IV009.
Clock resets integration	This section describes the clock reset integration options available with the Mali-IV009 ISP.
Running the simulation	This section provides information about how to run a testcase and compare the results.
Key implementation points	This section provides details of the key points you must consider for implementation of your design.
Floorplans	This section provides details of the floorplanning process in details and provides information about the various controls, constraints, inputs, outputs, and resources associated with the floorplanning process.

Section	Description
Equivalence checking	This section describes how to test the implementation using equivalence checking.
Sign-off	This section describes the verification criteria that you must satisfy before you can sign off the Arm Mali-IV009 ISP design.

Typographic conventions

Font/text type	What it means
italic	Introduces special terminology, denotes cross-references, and citations.
bold	Highlights interface elements, such as menu names. Denotes signal names. Also used for terms in descriptive lists, where appropriate.
`monospace`	Denotes text that you can enter at the keyboard, such as commands, file and program names, and source code.
`monospace italic`	Denotes arguments to monospace text where the argument is to be replaced by a specific value.
`monospace bold`	Denotes language keywords when used outside example code.
`<and>`	Encloses replaceable terms for assembler syntax where they appear in code or code fragments. For example: `MRC p15, 0, <Rd>, <CRn>, <CRm>, <Opcode_2>`

Timing diagrams

This figure explains the components used in timing diagrams. Variations, when they occur, have clear labels. You must not assume any timing information that is not explicit in the diagrams.

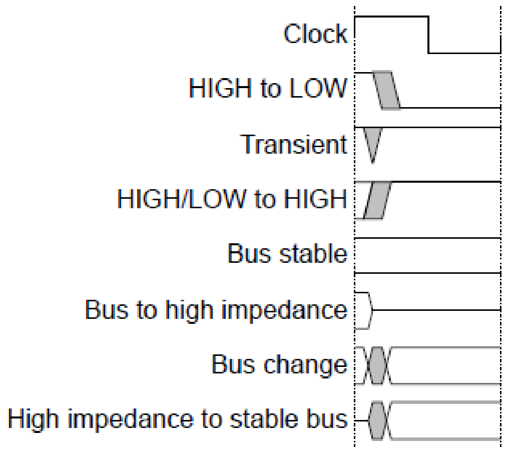


Figure 1. Key to timing diagram conventions

Shaded bus and signal areas are undefined, so the bus or signal can assume any value within the shaded area at that time. The actual level is unimportant and does not affect normal operation.

Signals

The signal conventions are:

Signal level

The level of an asserted signal depends on whether the signal is active-HIGH or active-LOW. Asserted means:

- HIGH for active-HIGH signals.
- LOW for active-LOW signals.

Lower-case n

At the start or end of a signal name denotes an active-LOW signal.

Additional reading

Information published by Arm and by third parties.

See https://developer.arm.com/ for access to Arm documentation.

Feedback on this product

If you have any comments or suggestions about this product, contact your supplier and give:

- The product name.
- The product revision or version.
- An explanation with as much information as you can provide. Include symptoms and diagnostic procedures if appropriate.

Feedback on content

If you have comments on content then send an e-mail to errata@arm.com. Give:

- The title.
- The number ARM-EPM-136047.
- The page number(s) to which your comments refer.
- A concise explanation of your comments.

Arm also welcomes general suggestions for additions and improvements.

Note: Arm tests the PDF only in Adobe Acrobat and Acrobat Reader, and cannot guarantee the quality of the represented document when used with any other PDF reader.

1 Introduction

This section describes the Mali-IV009 block diagram at a very high level. The block diagram shows the Arm IP and the various interface points to integrate this with customer IP.

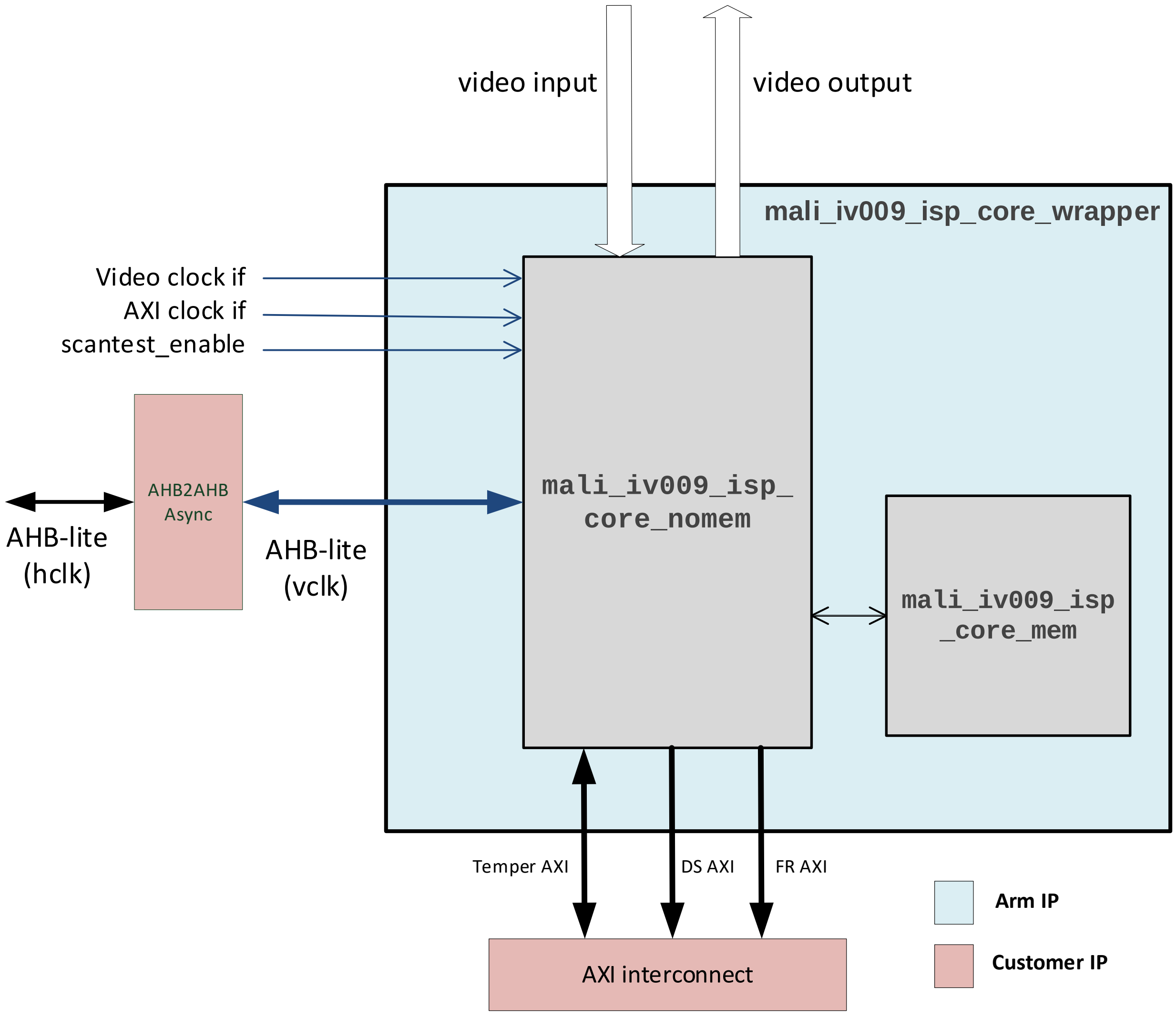
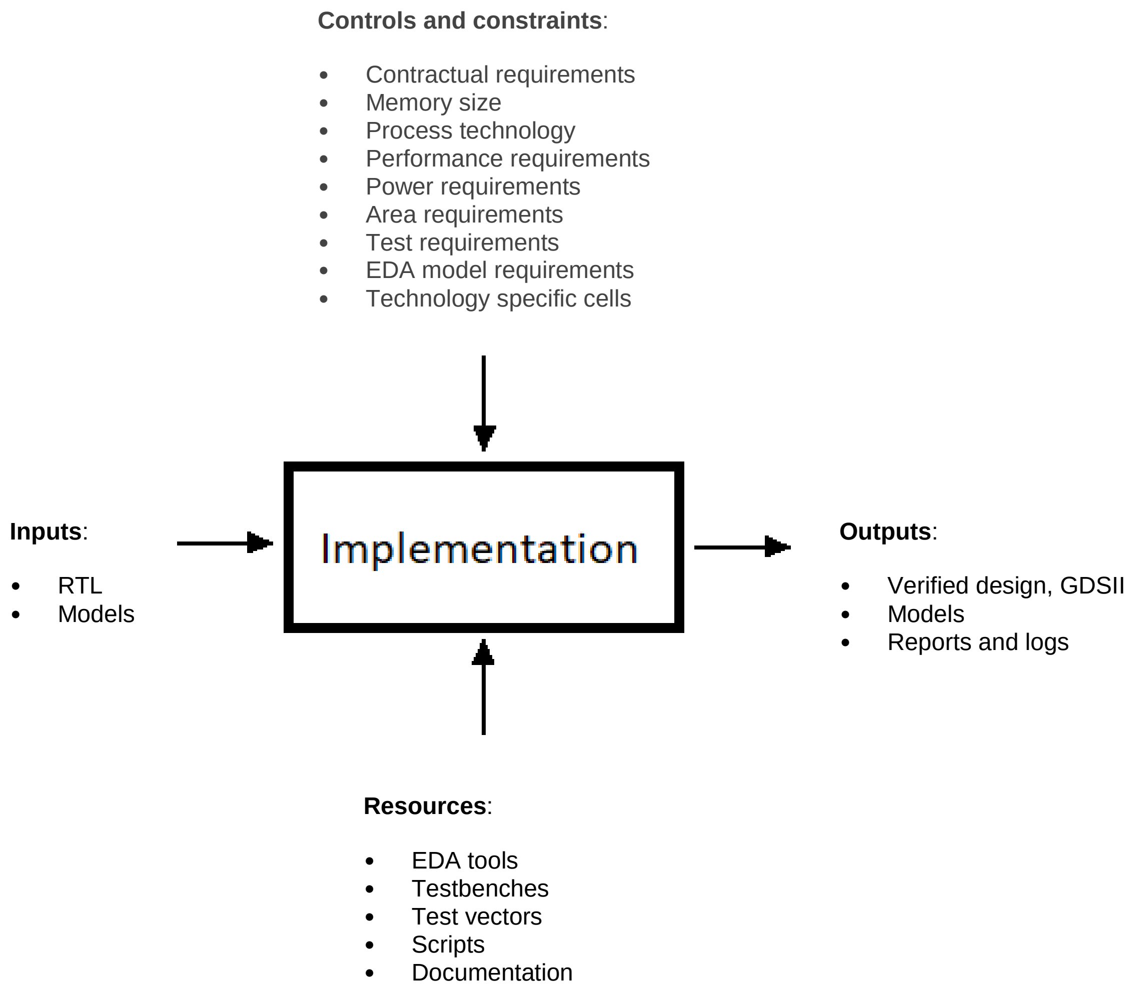


Figure 2. Mali-IV009 block diagram

2 The implementation process

The following figure shows the implementation process, including the top-level inputs, resources, controls and constraints, and outputs for the process.



2.1 Implementation process resources

This section describes the implementation process resources.

Certain resources apply to the implementation process.

- EDA tools
- Testbenches

- Test vectors
- Scripts
- Documentation

For the implementation process, you need suitable Electronic Design Automation (EDA) tools and compute resources.

2.2 Implementation process controls and constraints

These are the controls and constraints that apply to the implementation process:

- Contractual requirements.
- Memory size.
- Process technology.
- Performance requirements.
- Power requirements.
- Area requirements.
- Test requirements.
- EDA model requirements.

Note: The implementation must be in accordance with the contract, see Implementation obligations on page 7.

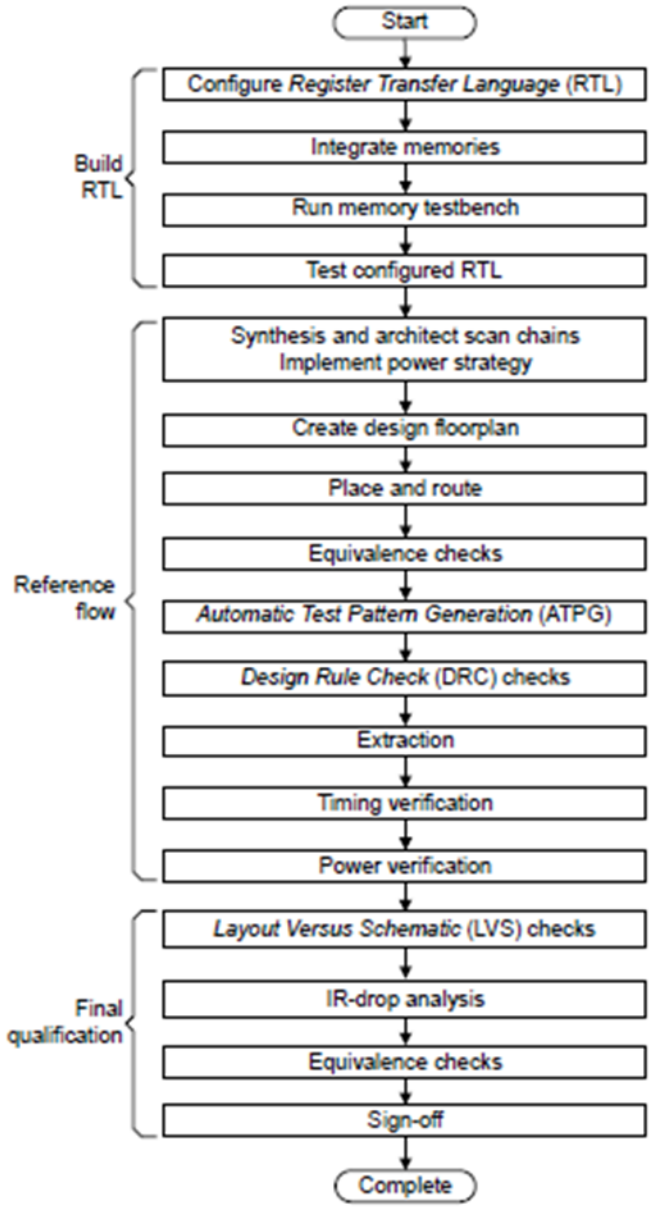
2.3 Implementation inputs

The deliverables that are inputs to the implementation process include:

- RTL
- Simulator models
- Documentation

2.4 Implementation process flow

The following figure shows the implementation process flow.



2.5 Implementation process output

These outputs are the controls and constraints that apply to the implementation process.

The outputs from the implementation process flow are:

- Logs and reports:
    - Implemented RTL validation logs and reports.
    - Synthesis logs and reports.
    - Post-layout Static Timing Analysis (STA) logs and reports.
    - Logs and reports showing logical equivalence of post-layout netlist with implemented RTL.
- Components:
    - Post-layout netlist
    - GDS-II
- Test:
    - ATPG vectors

3 Development tools

The following table lists the development tools that you will need during the configuration and integration of the Mali-IV009.

Purpose	Vendor	Tool
RTL simulator	Synopsys	VCS 2017.03-SP2
	Cadence	IUS 15.20.017
Miscellaneous	GNU	Make utility
		GCC 4.9.2 or later
		Llibstdc++ 3.4.20 or 3.4.21
Platform OS	Red Hat Linux	Enterprise 6
Scripting	Open source	Python 2.7.13
		NumPy 1.11.0
Implementation	Cadence	Genus 16.21.000
		Innovus 16.21.000
		Conformal 17.10.240
		QRC 16.10.000
		Tempus 16.23.000
	Synopsys	PrimeTime PX 2016.12
	Mentor Graphics	Tessent 2017.3

Table 1. Development tools

The following GNU utilities must be installed on your system:

- make.
- diff.

Note: You must use the GNU make utility specifically rather than any default make utility installed on your system.

The GNU make utility can exist in your UNIX system under a variety of names such as make, gmake, or gnumake.

To find the version of GNU make utility installed on your system:

1. Type the make name with the option –v.
2. Press Enter.

You must use version 3.81 or later of GNU make.

Note: ***See the Release Note for version information on the simulation and synthesis tools used in the testing of this product. If you use different design tools you must modify the configuration, scripts, and makefiles.***

4 Installation

Intellectual Property (IP) deliverables are delivered as one or more UNIX zipped tar files. The installation instructions cover the UNIX operating system only.

Installation involves:

- Downloading the deliverables
- Unpacking the deliverables
- Merging the deliverables

Additional information may also be available in the release email notification from Arm.

4.1 Downloading the deliverables

To download the deliverables:

1. Click on the Connect http links in the release email.
2. Click on **Add to Download** for each bundle or deliverable.
3. After all items have been selected for download, click the **download** button and wait for the transaction to be built.
   The window is then refreshed to show the size of the transaction, a checksum number and a link called "Download Now" at the bottom of the page.
4. Click the **Download Now** link.
5. Save the `arm-download-<transaction_id>.tgz` file to the target machine.

4.2 Unpacking the deliverables

To unpack the deliverable:

Use the GNU `gtar` utility to unpack it with the following Unix command:

```
% tar -xzf arm-download-<transaction_id>.tgz
```

For each download from the Arm Connect IP Delivery Server, the following two extra files are created:

- `ARM_DELIVERY_<transaction_id>.TXT`
- `ARM_MANIFEST_<transaction_id>.TXT`

where `<transaction_id>` is a unique delivery number. These files should be used to view the contents (parts or files) of the delivery or to investigate possible download corruption problems.

- **`ARM_DELIVERY_<transaction_id>.TXT`** lists the downloaded parts and the constituent parts of any downloaded bundle.
- **`ARM_MANIFEST_<transaction_id>.TXT`** contains a manifest of all the files included in the transaction, together with their checksums. The checksums provided are calculated using the RSA Data Security, Inc. MD5 Message-Digest Algorithm. The checksums can be used to verify the integrity of the data using the md5sum tool (and is part of the GNU textutils package) by running (in Unix):

  ```
  % md5sum --check ARM_MANIFEST_<transaction_id>.TXT
  ```

4.3 Merging the Deliverables

After unpacking the deliverables using GNU gtar, each bundle or separate deliverable will be contained in its own directory for example:

```
IV009-MN-22110-r0p0-00rel0/

IV009-DC-70021-r0p0-00rel0/
```

To merge the deliverables into a single installation directory:

Copy the contents of each supplied deliverable into the downloaded bundle directory so that it appears at the same directory level as shown in Figure 3.

For example, using the following Unix command:

```
% mkdir mali_iv009

% cp -r IV009-BU-40000-r0p0-00rel0/* mali_iv009

% cp -r IV009-BU-60000-r0p0-00rel0/* mali_iv009
```

4.4 Directory structure

Unpacking the deliverable creates the following directory structure on the target machine:

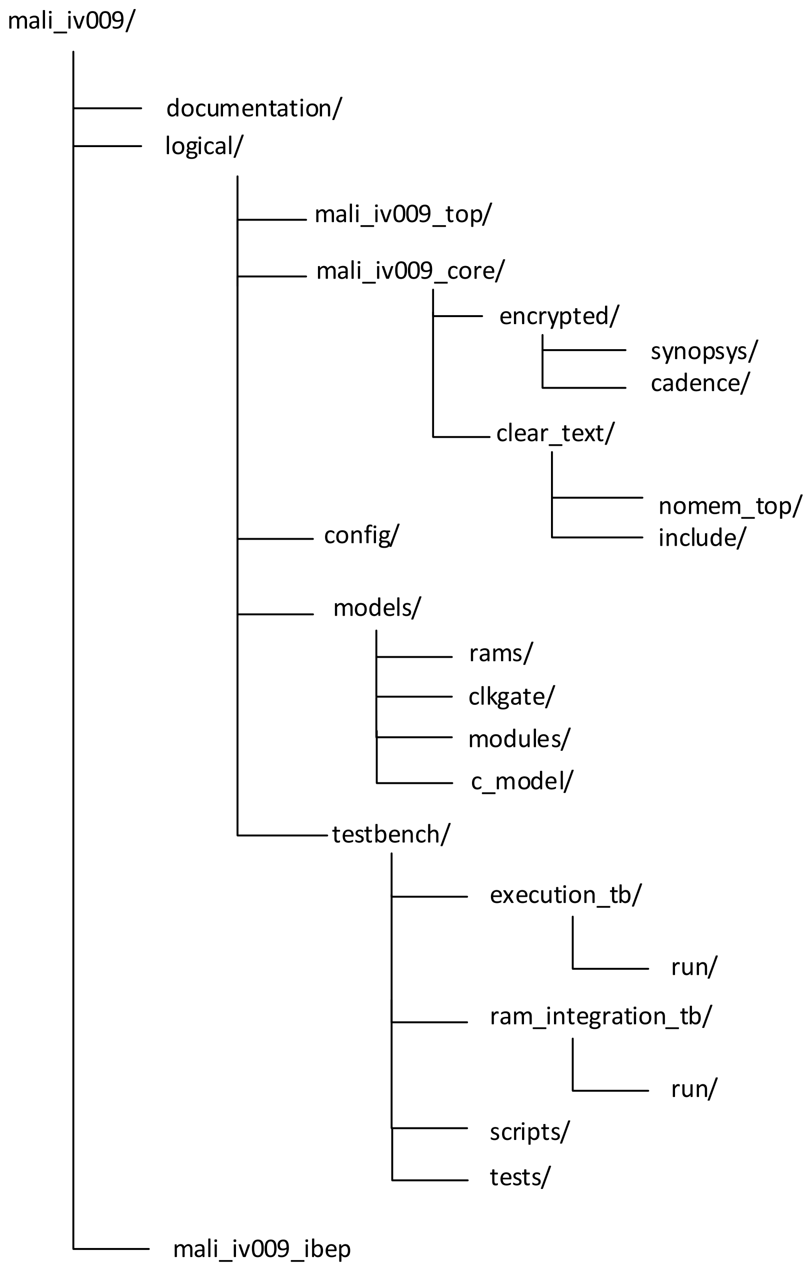


Figure 3. The Mali-IV009 directory structure

mali_iv009

This is the root directory of the package. You must configure this variable to store the fully qualified directory name where the source code is expanded.

The following table provides information about the various directories present under `mali_iv009`:

Directory	Description
`config`	This directory contains the top level configurable file `mali_iv009_top_global_param.vi`. This file contains default parameter values. You can modify this file if you need any configuration other than the default options. The configurability options are mentioned in section 6 Configuring the Mali-IV009.
`mali_iv009_top`	This folder contains the top-level wrapper containing the nomem and memory top.
`mali_iv009_core`	This folder contains all the encrypted and the clear text RTL files.
`encrypted`	This directory contains all the encrypted Verilog and VHDL files.
`synopsys`	This directory contains VHDL and Verilog files encrypted using synopsys synenc version D-2010.03-SP4 This directory also contains all the VHDL packages.
`cadence`	This directory contains VHDL and Verilog files encrypted using cadence ncprotect and also contains all VHDL packages.
`include`	This directory contains all the Verilog include files except the configurable include file. DO NOT modify the contents of the include directory.
`models`	This directory contains all the memory models, memory top level files and the manual clock gate behavioral model.
`rams`	This directory contains all the SRAM and DPRAM behavioral models and the memory toplevel file. These files are used in all simulations. You must replace all the files with the technology library memory models.
`clkgate`	This directory contains the behavioral clock gate cell model. You must replace this with the actual clock gate cell keeping the interface names same.
`c_model`	This directory contains the bit-exact executable C-model.
`ram_integration_tb`	This directory contains the RAM integration test bench.
`execution_tb`	This directory contains all the simulation testbench related files.
`scripts`	This directory contains all the scripts necessary to run the simulation.
`run`	This directory contains the Makefile.
`tests`	This directory contains all the test cases.

Directory	Description
`mali_iv009_ibep`	All the implementation related documents and the constraints are placed here.

Table 2. The Mali-IV009 directories

4.5 Source list

`mali_iv009/logical` contains the RTL source list.

The following table lists the ISP core files:

ISP core…	File name and location
ISP core nomem top-level file	`mali_iv009/logical/mali_iv009_core/clear_text/nomem_top/mali_iv009_isp_core_nomem.v`
ISP core Memory top-level file	`mali_iv009/logical/models/modules/mali_iv009_isp_core_mem.vhd` This file is for reference only. This file is used for all simulations.
ISP core top-level Wrapper file	`mali_iv009/logical/ mali_iv009_top/mali_iv009_isp_core_wrapper.v`

Table 3. ISP core files

The following table describes the various list files:

The list file…	Contains…
`mali_iv009_synopsys_vhdl.list`	The list of all synthesizable VHDL RTL files. All the files are encrypted using synopsys synenc.
`mali_iv009_synopsys_verilog.list`	The list of all synthesizable Verilog RTL files. This lists all the encrypted rtl and clear-text Verilog files. All the files are encrypted using synopsys synenc.
`mali_iv009_cadence_vhdl.list`	The list of all synthesizable VHDL RTL files. All the files are encrypted using cadence ncprotect.
`mali_iv009_cadence_verilog.list`	The list of all synthesizable Verilog RTL files. This lists all the encrypted rtl and clear-text Verilog files. All the files are encrypted using cadence ncprotect.
`mali_iv009_common_mem_vhdl.list`	The list of all clear-text memory models and the memory toplevel file.

Table 4. The RTL source list

5 Release package sanity check

After unpacking the deliverables you must do a release package sanity check.

This check involves:

- Compiling the RTL
- Running the RAM integration testbench

5.1 Compiling the RTL

The RTL files are encrypted using the Synopsys and Cadence encryption tools. Therefore, you can compile the RTL using VCS (Synopsys) and IUS (Cadence).

5.1.1 Compiling using VCS

The Synopsys encrypted files are located in the following directory:

```
mali_iv009/logical/mali_iv009_core/encrypted/synopsys/
```

To run the compilation using VCS:

1. Change the path to the run directory:

   ```
   Cd mali_iv009/logical/testbench/execution_tb/run
   ```

2. Type `make clean` and press **Enter**.
3. Type `make compile SIM=VCS` and press **Enter**.
4. Check for any compilation or elaboration errors.

5.1.2 Compiling using IUS

The Cadence encrypted files are located in the following directory:

```
mali_iv009/logical/mali_iv009_core/encrypted/cadence/
```

To run the compilation using IUS:

1. Change the path to the run directory:

   ```
   Cd mali_iv009/logical/testbench/execution_tb/run
   ```

2. Type `make clean` and press **Enter**.

3. Type `make compile SIM=IUS` and press **Enter**.
4. Check for any compilation or elaboration errors.

5.2 Running the RAM integration check

Once the release package is unpacked you must perform RAM integration test on the release package.

The single Makefile is placed in the `mali_iv009/logical/testbench/ram_integration_tb/run` folder.

The `ram_integration_tb/run` directory contains a Makefile.

The Makefile is common for both the IUS and the VCS. The Makefile has the following targets:

Makefile Target	Description
`help`	'make help' shows information about the targets (default)
`all`	Performs clean, compile and then runs the test
`compile`	compiles the design or testbench
`run`	executes the simulation for specific testcases
`clean`	removes the simulation and compile directories

The run commands are as below:

```
make <compile/run/all> SIM=<IUS/VCS>
```

5.2.1 Using VCS

The RAM integration test benches are located in the following directory:

```
mali_iv009/logical/testbench/ram_integration_tb
```

To run the test using VCS:

1. Change the path to the run directory:

   Cd `mali_iv009/logical/testbench/ram_integration_tb/run`

2. Type `make clean` and press **Enter**.

3. Type `make all SIM=VCS` and press **Enter**.
4. Check the `ram_int_synopsys_log.txt` file in the run directory to see if any compilation/elaboration errors have occurred.
5. Check the `sim_ram_tb_synopsys.log` file in the run directory to see the RAM integration test status.

5.2.2 Using IUS

To run the test using IUS:

1. Change the path to the run directory:

   ```
   Cd mali_iv009/logical/testbench/ram_integration_tb/run
   ```

2. Type `make clean` and press **Enter**.
3. Type `make all SIM=IUS` and press **Enter**.
4. Check the `ram_int_cadence_log.txt` file in the run directory to see if any compilation/elaboration errors have occurred.
5. Check the `sim_ram_tb_cadence.log` file in the run directory to see the RAM integration test status.

6 Configuring the Mali-IV009

Configuring the Mali-IV009 involves:

1. Modifying the configuration file.
2. Recompiling the RTL.

6.1 Modifying the configuration file

You can configure the Mali-IV009 as per your requirement by modifying the configuration file `mali_iv009_top_global_param.vi` present in the following directory:

```
mali_iv009/logical/mali_iv009_core/config
```

Note: This step is important and you must set correct configuration parameters based on your SoC requirements.

The following table describes the various configuration parameters:

Parameter name	Description	Supported Values
`AXI_WFIFO_AW`	Address width of all the DMA writer FIFOs. The FIFO depth is 2^AXI_WFIFO_AW.	Verified ONLY with 8 and 9. Default is 9.
`AXI_RFIFO_AW`	Address width of all the DMA reader FIFOs. The FIFO depth is 2^AXI_RFIFO_AW.	Verified ONLY with 8 and 9. Default is 9.
`AXI_WFIFO_UV_AW`	Address width of all the UV channel DMA reader FIFOs. The FIFO depth is 2^AXI_RFIFO_AW.	Verified ONLY with 8 and 9. Default is 8.
`DOWN_SCALED_PIPE_CHICKEN_OUT`	When this bit is set, the entire downscaled pipe will be removed from design. This includes all the configuration registers for the downscaled pipe. **Note:** the memory port will still remain and will be internally driven by safe values. For more information refer to the section 8.4.2 Down scale pipeline memory list.	0/1

Parameter name	Description	Supported Values
`DMSC_RGBIR_CHICKEN_OUT`	When this bit is set, the following conditions prevail: • The entire RGBIr demosaic module will be removed. • The Configuration registers will remain inside the design, and the software should not access it.	0/1
`ARM_ISP_LITE_ENABLE`	When this bit is set, the following configurations will be set inside the ISP: • CA correction module will be removed • Non-equidistant gamma and 3D LUT will be removed • Sinter version will be set to Sinter 2.5 But • Configuration registers will remain inside the ISP. • The configuration address space will be preserved and the software should not access that. • Memory interface signals will still remain. You should not connect any memory. For more information, refer to the section 8.4.3 ISP LITE memory list.	0/1

Table 5. Configuration parameters

An example FIFO depth vs latency analysis is shown below.

@600Mhz video, 600MHz AXIi clocks and 512 fifo depth the max latency tolerance of the FIFOs are:

FR/DS DMA running RGB32 format	: 3280ns
FR/DS UV DMA running P010 format	: 6560ns
TEMPER DMA running 20bits/pixel/master	: 5248ns

@600Mhz video, 600MHz AXI clocks and 256 fifo depth the max latency tolerance of the FIFOs are:

FR/DS DMA running RGB32 format	: 1573ns
FR/DS DMA running P010 format	: 3146ns
TEMPER DMA running 20bits/pixel/master	: 2517ns

6.2 Recompiling the RTL

Once you have modified the Config file with the correct required parameters you must recompile the RTL.

To recompile the RTL follow the steps listed in section 5.

7 Clock gate cell integration

The Mali-IV009 has manual clock gating cells for some of the modules.

Note: ***You must replace the provided behavioral model with your technical library clock gate cell.***

To replace the clock gate cell:

1. Locate the behavioral model of the clock gate cell in the following file: `logical/models/clkgate/NOSYN_GT.v`
2. Remove the internal logic.
3. Instantiate the integrated clock gate cell from your technology library inside this module.
4. Add the technology library clock gate cell file name in the appropriate file list.

8 Memory integration

This section describes the Mali-IV009 memory organization, and how to integrate memory blocks into the display processor.

Note: ***For successful configuration of the RTL you must:***
— Set up any configuration options.
— Replace behavioral memory models with vendor-specific memory macrocells.

Failure to complete all the necessary configuration can result in malfunction.

The `mali_iv009_isp_core_mem.vhd` design uses all the memory models. You must replace the behavioral simulation models with vendor or technology-specific memory macrocells.

8.1 Signal description table

The following table shows the signal names for any specific SRAM memory model. `<mem-name>` is the name of memory instance. The whole list of all the memories can be found in section 8.4 Memory list.

Signal name	Direction	Dimension	Description
`<mem_name>_wrdata`	Out	<>	Write data bus
`<mem_name>_rddata`	In	<>	Read data bus
`<mem_name>_addr`	Out	<>	Address bus for single port memories
`<mem_name>_wren`	Out	1	Write Strobe
`<mem_name>_rden`	Out	1	Read Strobe

For example, the following table shows the `cmd_queues` SRAM memory interface signals:

Signal name	Direction	Dimension
`cmd_queues_wren`	out	1
`cmd_queues_rden`	out	1
`cmd_queues_addr`	out	[7:0]
`cmd_queues_wrdata`	out	[31:0]
`cmd_queues_rddata`	in	[31:0]

Table 6. SRAM memory interface signals

Similarly, all the DPRAM memories follow a fixed suffix. For example, the `ds_scaler_lb_fifo_mem1` memory interface has the following signals:

Signal name	Direction	Dimension
`ds_scaler_lb_fifo_mem1_wraddr`	out	[12:0]
`ds_scaler_lb_fifo_mem1_wren`	out	1
`ds_scaler_lb_fifo_mem1_wrdata`	out	[35:0]
`ds_scaler_lb_fifo_mem1_rdaddr`	out	[12:0]
`ds_scaler_lb_fifo_mem1_rden`	out	1
`ds_scaler_lb_fifo_mem1_rddata`	In	[35:0]

Table 7. DPRAM memory interface signals

8.2 Timing diagram

The following timing diagram shows the SRAM interface signal timing relationship.

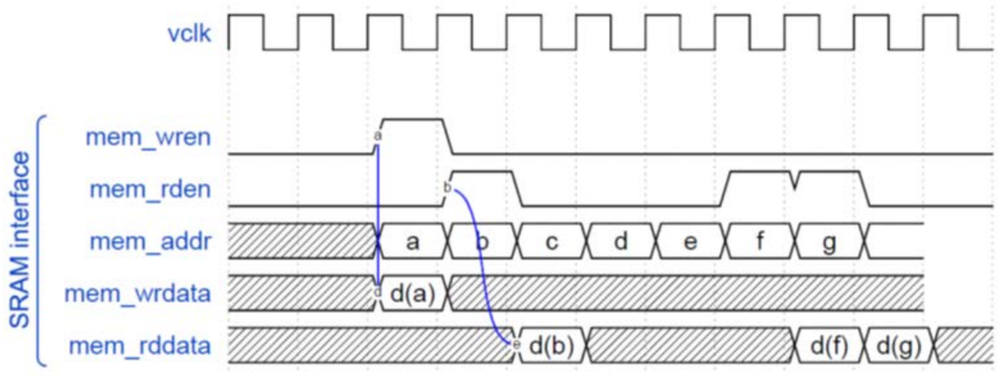


Table 8. SRAM interface signal timing relationship

8.3 Multi-channel memory

When the SRAM is multi-channel, then the specification of the SRAM is duplicated according to the number of channels. In this case data busses and address busses are concatenated:

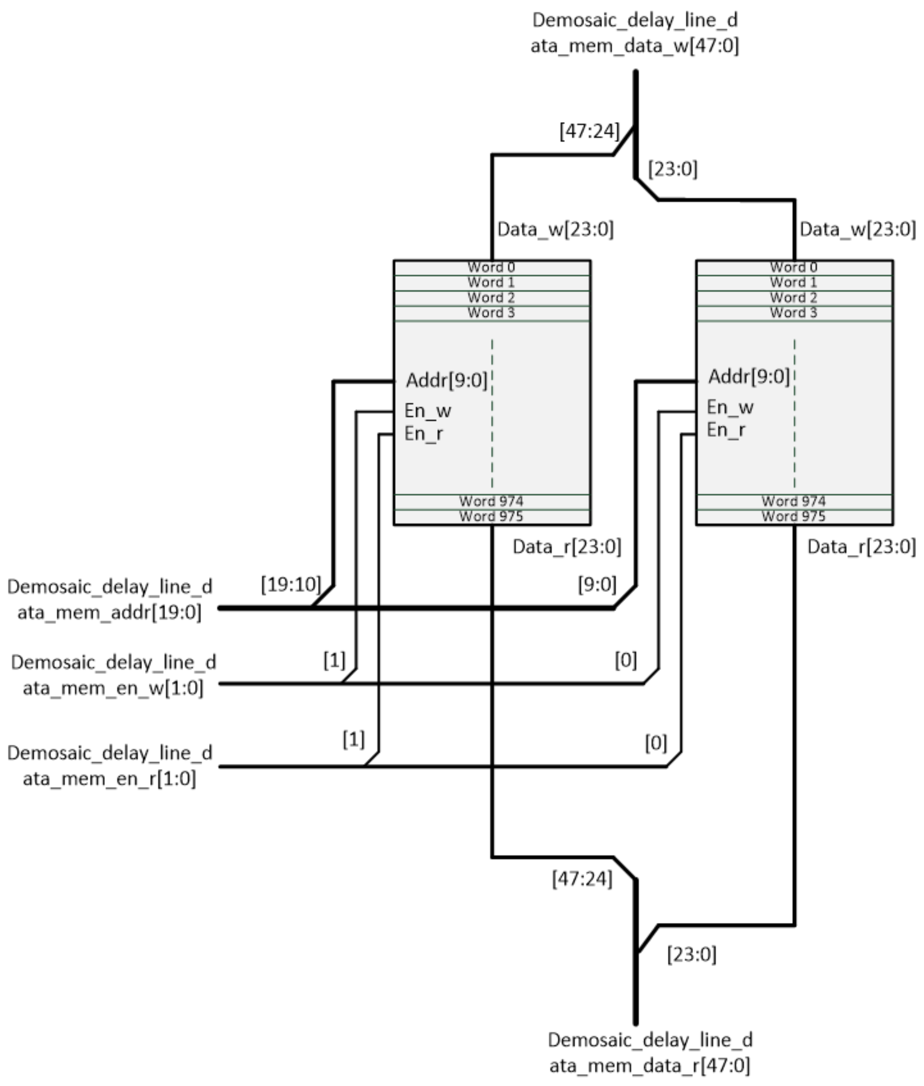

Demosaic_delay_line_data_mem_data_w[47:0]
[47:24]
[23:0]
Data_w[23:0]
Data_w[23:0]
Word 0
Word 1
Word 2
Word 3
Addr[9:0]
En_w
En_r
Word 974
Word 975
Data_r[23:0]
Data_r[23:0]
Demosaic_delay_line_data_mem_addr[19:0]
[19:10]
[9:0]
Demosaic_delay_line_data_mem_en_w[1:0]
[1]
[0]
Demosaic_delay_line_data_mem_en_r[1:0]
[1]
[0]
[47:24]
[23:0]
Demosaic_delay_line_data_mem_data_r[47:0]

Figure 4. Multi-channel memory connectivity

8.4 Memory list

This section lists all the memories used in the Mali-IV009. Every memory follows a fixed signal naming convention.

8.4.1 ISP common memories

The following table lists the memories that are common to all ISP configurations. (For configuration details, refer to section 6 Configuring the Mali-IV009):

Memory name	Mem type	clock	Data width	Addr width	words	channels
`cmd_queues`	sram	vclk	32	8	256	1
`mirror_mem`	sram	vclk	24	12	2320	1
`frame_stitch_dly1_mem`	sram	vclk	24	12	2320	3
`frame_stitch_dly2_mem`	sram	vclk	24	12	2320	3
`frame_stitch_dly3_mem`	sram	vclk	48	12	2320	4
`frame_stitch_lc1_mem`	sram	vclk	68	11	1168	1
`frame_stitch_lc2_mem`	sram	vclk	68	11	1168	1
`decompander0_ping_mem2`	sram	vclk	20	5	32	1
`decompander1_ping_mem2`	sram	vclk	20	8	256	1
`decompander0_pong_mem2`	sram	vclk	20	5	32	1
`decompander1_pong_mem2`	sram	vclk	20	8	256	1
`decompander0_ping_mem`	sram	vclk	20	5	32	1
`decompander0_pong_mem`	sram	vclk	20	5	32	1
`decompander1_ping_mem`	sram	vclk	20	8	256	1
`decompander1_pong_mem`	sram	vclk	20	8	256	1
`rfe_dly_mem`	sram	vclk	36	12	2320	4
`dpc_ping_mem`	sram	vclk	25	12	4096	1
`dpc_pong_mem`	sram	vclk	25	12	4096	1
`temper_delay_line_mem`	sram	vclk	128	12	2320	2
`temper_fb_msb_wfifo_mem`	dpram	clk_r=aclk clk_w=vclk	128	9	512	1
`temper_fb_msb_rfifo_mem`	dpram	clk_r=vclk clk_w=aclk	128	9	512	1
`temper_fb_lsb_wfifo_mem`	dpram	clk_r=aclk clk_w=vclk	128	9	512	1
`temper_fb_lsb_rfifo_mem`	dpram	clk_r=vclk clk_w=aclk	128	9	512	1
`mesh_shading_ping_mem`	sram	vclk	32	12	4096	1
`mesh_shading_pong_mem`	sram	vclk	32	12	4096	1
`radial_shading_ping_mem`	sram	vclk	16	9	512	1
`radial_shading_ping_mem2`	sram	vclk	16	9	512	1
`radial_shading_pong_mem`	sram	vclk	16	9	512	1
`radial_shading_pong_mem2`	sram	vclk	16	9	512	1
`iridix_mem`	sram	vclk	16	11	2048	4
`iridix_mem2`	sram	vclk	16	13	8192	4
`iridix_rp_ping_mem`	sram	vclk	20	7	128	1
`iridix_rp_ping_mem2`	sram	vclk	20	7	128	1
`iridix_rp_pong_mem`	sram	vclk	20	7	128	1

Memory name	Mem type	clock	Data width	Addr width	words	channels
iridix_rp_pong_mem2	sram	vclk	20	7	128	1
iridix_fp1_ping_mem	sram	vclk	20	7	128	1
iridix_fp1_ping_mem2	sram	vclk	20	7	128	1
iridix_fp1_pong_mem	sram	vclk	20	7	128	1
iridix_fp1_pong_mem2	sram	vclk	20	7	128	1
iridix_fp2_ping_mem	sram	vclk	20	7	128	1
iridix_fp2_ping_mem2	sram	vclk	20	7	128	1
iridix_fp2_pong_mem	sram	vclk	20	7	128	1
iridix_fp2_pong_mem2	sram	vclk	20	7	128	1
iridix_delay_line_data_mem	sram	vclk	40	12	2320	2
demosaic_dly_mem	sram	vclk	24	12	2320	8
uv_mean_rfilter_mem	sram	vclk	58	12	2320	1
uv_var_rfilter_mem	sram	vclk	58	12	2320	1
seg_mean_rfilter_mem	sram	vclk	58	12	2320	1
lut3d_mem	sram	vclk	30	7	125	8
lumvar_scratch_mem	sram	vclk	70	5	32	1
lumvar_stats_ping_mem	sram	vclk	22	9	512	1
lumvar_stats_pong_mem	sram	vclk	22	9	512	1
aexp_scratch_mem	sram	vclk	45	8	132	1
awb_scratch_mem	sram	vclk	39	8	132	1
af_scratch_mem	sram	vclk	71	8	132	1
af_delay_line_mem	sram	vclk	24	12	2320	2
metering_stats_ping_mem	sram	vclk	32	13	6576	1
metering_stats_pong_mem	sram	vclk	32	13	6576	1
aexp_hist_stats_mem	sram	vclk	27	10	1024	1
aexp_hist_stats_mem2	sram	vclk	16	10	1024	1
ihist_stats_mem	sram	vclk	27	10	1024	1
ihist_stats_mem2	sram	vclk	16	10	1024	1
fr_sharpen_dly1_mem	sram	vclk	60	12	2320	2
fr_sharpen_dly2_mem	sram	vclk	20	12	2320	2
fr_cs_conv_dly_mem	sram	vclk	20	12	2320	1
fr_dma_writer_wfifo_mem	dpram	clk_r=aclk clk_w=vclk	128	9	512	1
fr_uv_dma_writer_wfifo_mem	dpram	clk_r=aclk clk_w=vclk	128	8	256	1

Table 9. Common ISP memories

8.4.2 Down scale pipeline memory list

This memory list must be instantiated if the parameter `DOWN_SCALED_PIPE_CHICKEN_OUT` is set to 0 (zero) in the configuration file `mali_iv009_top_global_param.vi` present in the following directory:

`mali_iv009/logical/mali_iv009_core/config`

Note: ***If the parameter*** `DOWN_SCALED_PIPE_CHICKEN_OUT` ***is set to 1, the memories must not be instantiated and*** `<mem-name>_rddata` ***should be tied to 0.***

Memory name	Mem type	clock	Data width	Addr width	words	channels
`ds_scaler_hfilt_coefmem`	sram	vclk	56	9	256	1
`ds_scaler_vfilt_coefmem`	sram	vclk	56	9	256	1
`ds_scaler_vfilt_linemem1`	sram	vclk	72	12	2048	6
`ds_scaler_lb_fifo_mem1`	dpram	clk_r=vclk clk_w=vclk	36	13	8192	1
`ds_sharpen_dly1_mem`	sram	vclk	60	12	2064	2
`ds_sharpen_dly2_mem`	sram	vclk	20	12	2064	2
`ds_cs_conv_dly_mem`	sram	vclk	20	12	2064	1
`ds_dma_writer_wfifo_mem`	dpram	clk_r=aclk clk_w=vclk	128	9	512	1
`ds_uv_dma_writer_wfifo_mem`	dpram	clk_r=aclk clk_w=vclk	128	9	512	1

Table 10. Down scale pipeline memory list

Note: ***The*** *`ds_scaler_hfilt_coefmem`* ***and the*** *`ds_scaler_vfilt_coefmem`* ***addresses are 9 bits to hold the 16-sets of coefficients. But IV009 has restricted it to 8-sets. So only 256 locations are needed. Hence the MSB of the address buses will be unused.***

8.4.3 ISP LITE memory list

The following memory list must be instantiated if the parameter `ARM_ISP_LITE_ENABLE` is set to 0 (zero) in the configuration file `mali_iv009_top_global_param.vi` present in the following directory:

`mali_iv009/logical/mali_iv009_core/config`

Note: `sinter3_delay_line_mem` ***and*** `sinter2p5_delay_line_mem` ***described in the following tables are mutually exclusive. Please make sure that only 1 memory is instantiated based on the `ARM_ISP_LITE_ENABLE` configuration.***

Memory name	Mem type	clock	Data width	Addr width	words	channels
`sinter3_delay_line_mem`	sram	vclk	36	12	2320	32
`ca_correction_delay_line_mem`	sram	vclk	32	12	2320	12
`ca_correction_filter_ping_mem`	sram	vclk	32	5	32	1
`ca_correction_filter_pong_mem`	sram	vclk	32	5	32	1
`ca_correction_mesh_ping_mem`	sram	vclk	32	12	4096	1
`ca_correction_mesh_pong_mem`	sram	vclk	32	12	4096	1
`lut3d_mem`	sram	vclk	30	7	125	8

Table 11. ISP LITE memory list

The following memory list must be instantiated if the parameter `ARM_ISP_LITE_ENABLE` is set to 1 (one) in the configuration file `mali_iv009_top_global_param.vi` present in the following directory:

`mali_iv009/logical/mali_iv009_core/config`

Memory name	Mem type	clock	Data width	Addr width	words	channels
`sinter2p5_delay_line_mem`	sram	vclk	36	12	2320	16

Note: If a memory is not instantiated, you must connect the read data bus to 0.

8.5 Delay line address limiting

Where memories are used as line delays it is common for part of the address range to be unused. In these cases, it is possible to prevent the ISP from accessing out of range memory by configuring a memory address limit. This limit is configured by setting the appropriate value to `max_addr_dly_line_fr` pand `max_addr_dly_line_ds` ports.

8.5.1 Depth of the delay line

The depth of the delay line memory is depicted by the equation:

```
ceiling(MAX_FRAME_WIDTH/2)+16
```

For example:

The default depth of the memory `frame_stitch_dly1_mem` is:

`Ceiling(4608/2)+16 = 2320` for 16MP resolution.

Similarly, the depth of the same memory for 2MP resolution will be:

`Ceiling(1920/2)+16 = 976`

8.5.2 Setting the delay line value

You must set the `max_addr_dly_line_fr` value to:

`ceiling(MAX_FRAME_WIDTH/2)+16) – 1`

The following table list the memories for which you should set the memory address limit through `max_addr_dly_line_fr`.

Memory name
mirror_mem
frame_stitch_dly1_mem
frame_stitch_dly2_mem
frame_stitch_dly3_mem
rfe_dly_mem
sinter2p5_delay_line_mem
sinter3_delay_line_mem
temper_delay_line_mem
ca_correction_delay_line_mem
iridix_delay_line_data_mem
demosaic_dly_mem
uv_mean_rfilter_mem
uv_var_rfilter_mem
seg_mean_rfilter_mem
af_delay_line_mem
fr_sharpen_dly1_mem
fr_sharpen_dly2_mem
fr_cs_conv_dly_mem

Table 12. Words configurable memory list 1

For the following memory list, the depth of the memory must be limited to:

```
ceiling(MAX_FRAME_WIDTH/4)+16
```

Memory name
`frame_stitch_lc1_mem`
`frame_stitch_lc2_mem`

Table 13. Words configurable memory list 2

You must set the `max_addr_dly_line_ds` value to:

```
ceiling(MAX_FRAME_WIDTH_POST_SCALER/2)+16) – 1
```

The following table list the memories for which you should set the memory address limit through the max_addr_dly_line_ds.

Memory name
`ds_sharpen_dly1_mem`
`ds_sharpen_dly2_mem`
`ds_cs_conv_dly_mem`

Table 14. Words configurable memory list 3

9 RAM integration test bench

This section provides information related to running the RAM integration test bench.

Once the memory integration step is complete, you must run the RAM integration testbench as described in section 5.2 Running the RAM integration check.

- To use the Mali-IV009 RAM integration testbench, the interface signal names must be the same and all the memory ports must be present.
- You must modify the `mali_iv009_common_mem_vhdl.list` to delete the behavioral memory models and add the technology library memory models.
- There are 2 clocks defined in the test-bench. You should change either the vclk or aclk periods so that the relationships **aclk > vclk** and **aclk <= vclk** are verified.

  For example:

  constant period_vclk : integer := 10;

  constant period_aclk : integer := 8;

Note: Whenever the memory list is updated, you must run the RAM integration testbench as described in section 5.2 Running the RAM integration check.

The following figure shows the RAM integration test bench:

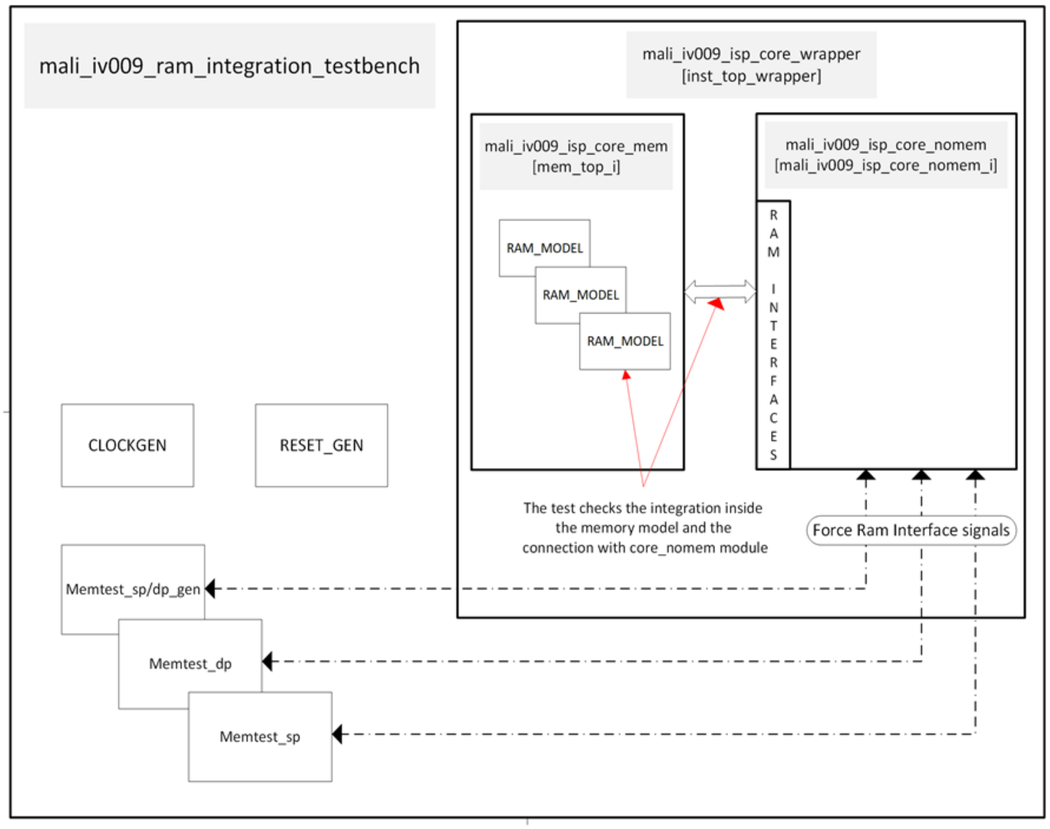

mali_iv009_ram_integration_testbench
mali_iv009_isp_core_wrapper [inst_top_wrapper]
mali_iv009_isp_core_mem [mem_top_i]
mali_iv009_isp_core_nomem [mali_iv009_isp_core_nomem_i]
RAM_MODEL
RAM_MODEL
RAM_MODEL
RAM INTERFACES
CLOCKGEN
RESET_GEN
The test checks the integration inside the memory model and the connection with core_nomem module
Force Ram Interface signals
Memtest_sp/dp_gen
Memtest_dp
Memtest_sp

Figure 5. RAM integration test bench

The ram integration test bench is placed in:

`mali_iv009/logical/testbench/ram_integration_tb/mali_iv009_ram_integration_testbench.vhd`

To use the Mali-IV009 RAM Integration Testbench you must ensure the following:

- The interface signal names and instance names must be the same.
- All memory ports must be present.
- Generic values must be configured according to the desired configuration.
  The following generics MUST be modified based on the ISP configuration and memory configurations.
  - `max_addr_dly_line_fr`
  - `max_addr_dly_line_ds`
  - `down_scaled_pipe_chicken_out`

  - `arm_isp_lite_enable`
  - `axi_wfifo_aw`
  - `axi_rfifo_aw`
  - `axi_wfifo_uv_aw`

For more information regarding these parameters, refer to section 6.1 Modifying the configuration file.

9.1 RAM integration testbench tests

This section provides information about the tests that the RAM integration testbench performs.

Each test case:

- performs 8 tests on each single port memory.
- performs 7 tests on each double port memory.
- checks if the values read from the memory matches with the expected values.

The RAM integration testbench performs the following tests:

- Write all ones to each memory location and read back after all writes have been completed.
- Write all zeros to each memory location and read back after all writes have been completed.
- Write and Read increasing values to all memory locations consecutively for single port memories.
  For double port memories, the Read happens one line behind each Write.
- Write and Read decreasing values to all memory locations consecutively for single port memories.
  For double port memories, the Read happens one line behind each Write.
- Write first the same patterns as step 3 to the whole memory and then read back from them. This step stimulates the chip enable signal connection trying to write data in memory disabling it.
- Write random data into the memory and then read back.
- Randomize the address signal.
- For single port memories, check connectivity for multichannel memories.

9.2 RAM integration test failures

Table 15. lists the most common failure sequences and the possible causes of test failures.

If a failure occurs, check both the connections inside the memory model and the connections between the RAM and the RAM interfaces of the core module.

Note: ***This is not a comprehensive and exact list. It is just a guideline and there may be different issues associated with test failures.***

Failure test number	Possible cause of failure
1	Problem with data write or read signal of the memory. Check if any wires are stuck to ‘1’.
2	Problem with data write or read signal of the memory. Check if any wires is stuck to ‘0’.
3	Problem with address connectivity or write data signal.
4	Problem with address connectivity or write data signal.
5	Problem with chip enable signal.
6	Problem with address connectivity or read data signal.
7	Problem with address connectivity.
8	The same wires of a signal are connected to more than one instance of this multichannel memory.
1,2,5,5,6,7	Check if write enable signal is stuck to ‘1’.
3,4,6,7	Check Addresses connection.
1,3,4,5,5,6,7	Check write data signal connection.
2,3,4,5,5,6,7	Check write data signal connection.
5 (just one time)	Check chip enable connection.

Table 15. Failure analysis guidelines

10 Mali-IV009 top interfaces

This section describes the Mali-IV009 top interface signals and their properties.

10.1 Video clock interface

This is the pixel clock interface. All the video data is sampled at the positive edge of this clock.

Signal name	Direction	Dimension	Description
`vclk`	In	1	Video pixel clock.
`rstn`	In	1	Active low reset in vclk domain.
`vcke`	In	1	Global Clock enable signal. This signal should only be used to stall the ISP globally.

10.2 AXI clock interface

Signal name	Direction	Dimension	Description
`aclk`	In	1	AXI clock. All the AXI signals are sampled at the positive edge of this clock.
`aresetn`	In	1	Active low reset in the aclk domain.

10.3 AHB lite slave interface

Clock domain: vclk

The ISP core uses the standard AHB-lite slave interface with certain restrictions which are mentioned in the following table:

Signal name	Direction	Dimension	Description
`haddr`	In	[17:0]	The 17-bit system address bus. This must be the offset address starting with 0 as the base address.
`hburst`	In	[2:0]	The burst type indicates if the transfer is a single transfer or forms part of a burst. This is ignored internally as each individual transaction is converted to its equivalent SRAM transaction.
`hmastlock`	In	1	When HIGH, this signal indicates that the current transfer is part of a locked sequence. But in Mali-IV009 this is ignored internally.
`hprot`	in	[3:0]	This signal exists in the interface for

Signal name	Direction	Dimension	Description
			completeness. If a read operation follows immediately after a write operation, the write address and write data are stored in an internal buffer, and the SRAM carries out the read operation first. The stalled write transfer is carried out when the AHB interface is idle, or when there is a write transfer. A merging of read data between the internal buffers and the read data from SRAM is carried out automatically by the interface module, when: • a read operation follows immediately after a write operation to the same address • a sequence of read operations follows immediately after a write operation with any of the read transfers using the same address. The merging processing uses the valid status of the internal buffer byte and ensures the read data that returns to the bus master is up-to-date. This process occurs invisibly and does not result in any wait states.
`hready`	In	1	This is the input hready signal.
`hselahb`	In	1	When the slave is initially selected, it must also monitor the status of HREADY to ensure that the previous bus transfer has completed, before it responds to the current transfer. The hselahb signal is a combinatorial decode of the address bus.
`hsize`	In	[2:0]	Mali-IV009 supports only 2 (32-bit word access)
`htrans`	In	[1:0]	Indicates the transfer type of the current transfer. This can be: • IDLE • BUSY • NONSEQUENTIAL • SEQUENTIAL.
`hwdata`	in	[31:0]	The write data bus transfers data from the master to the slaves during write operations. A minimum data bus width of 32 bits is recommended. However, this can be extended to enable higher bandwidth

Signal name	Direction	Dimension	Description
			operation.
hwrite	In	1	Indicates the transfer direction. When HIGH, this signal indicates a write transfer and when LOW it indicates a read transfer. It has the same timing as the address signals, however, it must remain constant throughout a burst transfer.
hrdata	Out	[31:0]	During read operations, the read data bus transfers data from the selected slave to the multiplexer. The multiplexer then transfers the data to the master. A minimum data bus width of 32 bits is recommended. However, this can be extended to enable higher bandwidth operation.
hreadyout	Out	1	When HIGH, the HREADYOUT signal indicates that a transfer has finished on the bus. This signal can be driven LOW to extend a transfer.
hresp	Out	1	The Mali-IV009 supports only an OKAY response.

10.4 AXI master interfaces

Clock domain: aclk

The Mali-IV009 has four AXI-3 Interfaces, three writes and one read.

Note: The Mali-IV009 AXI masters do not support the write data interleaving feature.

10.4.1 Full resolution axi_master_write interface

Signal name	Direction	Dimension
fr_video_out_axi_awid	out	[5:0]
fr_video_out_axi_awaddr	out	[31:0]
fr_video_out_axi_awlen	out	[3:0]
fr_video_out_axi_awsize	out	[2:0]
fr_video_out_axi_awburst	out	[1:0]
fr_video_out_axi_awlock	out	[1:0]
fr_video_out_axi_awcache	out	[3:0]
fr_video_out_axi_awprot	out	[2:0]
fr_video_out_axi_awvalid	out	1
fr_video_out_axi_awready	in	1

Signal name	Direction	Dimension
`fr_video_out_axi_wid`	out	[5:0]
`fr_video_out_axi_wdata`	out	[127:0]
`fr_video_out_axi_wstrb`	out	[15:0]
`fr_video_out_axi_wlast`	out	1
`fr_video_out_axi_wvalid`	out	1
`fr_video_out_axi_wready`	in	1
`fr_video_out_axi_bid`	in	[5:0]
`fr_video_out_axi_bresp`	in	[1:0]
`fr_video_out_axi_bvalid`	in	1
`fr_video_out_axi_bready`	out	1

10.4.2 Downscaled resolution axi_master_write interface

Signal name	Direction	Dimension
`ds_video_out_axi_awid`	out	[5:0]
`ds_video_out_axi_awaddr`	out	[31:0]
`ds_video_out_axi_awlen`	out	[3:0]
`ds_video_out_axi_awsize`	out	[2:0]
`ds_video_out_axi_awburst`	out	[1:0]
`ds_video_out_axi_awlock`	out	[1:0]
`ds_video_out_axi_awcache`	out	[3:0]
`ds_video_out_axi_awprot`	out	[2:0]
`ds_video_out_axi_awvalid`	out	1
`ds_video_out_axi_awready`	in	1
`ds_video_out_axi_wid`	out	[5:0]
`ds_video_out_axi_wdata`	out	[127:0]
`ds_video_out_axi_wstrb`	out	[15:0]
`ds_video_out_axi_wlast`	out	1
`ds_video_out_axi_wvalid`	out	1
`ds_video_out_axi_wready`	in	1
`ds_video_out_axi_bid`	in	[5:0]
`ds_video_out_axi_bresp`	in	[1:0]
`ds_video_out_axi_bvalid`	in	1
`ds_video_out_axi_bready`	out	1

10.4.3 temper_axi_master_write

Signal name	Direction	Dimension
`temper_axi_write_awid`	out	[3:0]
`temper_axi_write_awaddr`	out	[31:0]

Signal name	Direction	Dimension
`temper_axi_write_awlen`	out	[3:0]
`temper_axi_write_awsize`	out	[2:0]
`temper_axi_write_awburst`	out	[1:0]
`temper_axi_write_awlock`	out	[1:0]
`temper_axi_write_awcache`	out	[3:0]
`temper_axi_write_awprot`	out	[2:0]
`temper_axi_write_awvalid`	out	1
`temper_axi_write_awready`	in	1
`temper_axi_write_wid`	out	[3:0]
`temper_axi_write_wdata`	out	[127:0]
`temper_axi_write_wstrb`	out	[15:0]
`temper_axi_write_wlast`	out	1
`temper_axi_write_wvalid`	out	1
`temper_axi_write_wready`	in	1
`temper_axi_write_bid`	in	[3:0]
`temper_axi_write_bresp`	in	[1:0]
`temper_axi_write_bvalid`	in	1
`temper_axi_write_bready`	out	1

10.4.4 temper_axi_master_read

Signal name	Direction	Dimension
`temper_axi_read_arid`	out	[3:0]
`temper_axi_read_araddr`	out	[31:0]
`temper_axi_read_arlen`	out	[3:0]
`temper_axi_read_arsize`	out	[2:0]
`temper_axi_read_arburst`	out	[1:0]
`temper_axi_read_arlock`	out	[1:0]
`temper_axi_read_arcache`	out	[3:0]
`temper_axi_read_arprot`	out	[2:0]
`temper_axi_read_arvalid`	out	1
`temper_axi_read_arready`	in	1
`temper_axi_read_rid`	in	[3:0]
`temper_axi_read_rdata`	in	[127:0]
`temper_axi_read_rresp`	in	[1:0]
`temper_axi_read_rlast`	in	1
`temper_axi_read_rvalid`	in	1
`temper_axi_read_rready`	out	1

10.5Video interface

This section provides information about the video input and output interfaces.

Clock domain: vclk

10.5.1 Video input interface

The interface supports continuous lines of data. Every rising edge of `hvalid_i` signal is interpreted as line start condition. If the data source is likely to stall in the middle of the line (for example, data is coming from different clock domain with slower clock), then the FIFO should be instantiated (outside the ISP) to provide for uninterrupted lines. The ISP is not sensitive to horizontal and vertical blanking provided the minimal blanking requirements are met.

10.5.1.1 Video Input (RAW) signal description table

Signal name	Direction	Dimension	Description
`hvalid_i`	In	1	'Horizontal valid' signal: '1' means image data is present; '0' means blanking between lines and between frames
`vvalid_i`	In	1	'Vertical valid' signal: '1' means frame is active; '0' means vertical blanking between frames.
`vdata<0,1,2,3>_i`	In	[19:0]	Pixel data, MSB aligned when hvalid_in='1' and vvalid_in='1'; ignored if hvalid_in='0' or vvalid_in='0' DOL: vdata0_i → very short exposures vdata1_i → short exposures vdata2_i → medium exposures vdata3_i → long exposures OV10640 3:2 mux mode: (see table below) OV2770 16-bit linear + 12 bit VS mode: (see table below)

OV10640 3:2 mux mode:

	[19]	[18]	[17:8]	[7:0]
vdata0_i	X	select	L/Vs	XXX
vdata1_i	X	X	S	XXX
vdata2_i	X	X	XXX	XXX
vdata3_i	X	X	XXX	XXX

OV2770 16-bit linear + 12 bit VS mode:

	[19:8]	[7:4]	[3:0]
vdata0_i	16-bit linear		X

Signal name	Direction	Dimension	Description

vdata1_i	BS	X	X
vdata2_i	X	X	X
vdata3_i	X	X	X

OV2770 12-bit companded + 12 bit VS mode:

	[19:8]	[7:0]
vdata0_i	12-bit companded	X
vdata1_i	VS	X
vdata2_i	X	X
vdata3_i	X	X

All other modes:

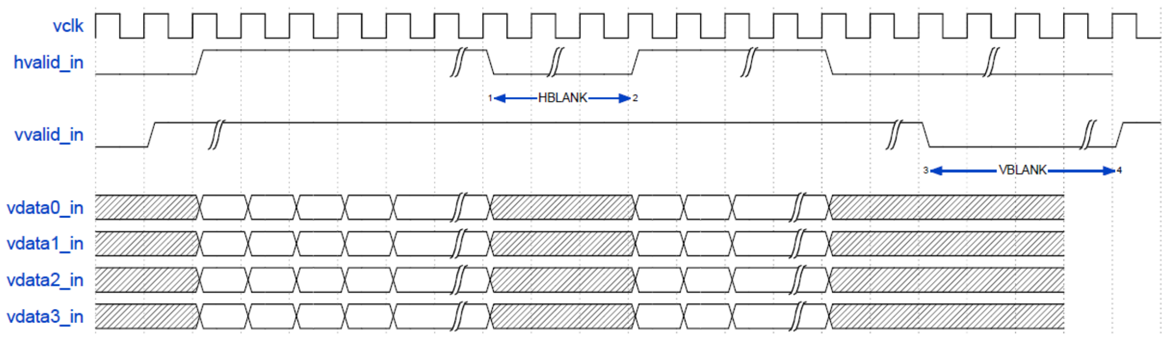
	[19:n]	[n-1:0]
vdata0_i	MSB aligned	X
vdata1_i	X	X
vdata2_i	X	X
vdata3_i	X	X

RGB/YUV bypass:

	[19:n]	[n-1:0]
vdata0_i	Y/R (MSB aligned)	X
vdata1_i	U/G (MSB aligned)	X
vdata2_i	V/B (MSB aligned)	X
vdata3_i	X	X

10.5.1.2 Video Input (RAW) Timing Diagram

The following diagram shows the timing relationship between the input signals.



10.5.2 Streaming video output interface

The Mali-IV009 has two streaming output channels, one of them is full resolution and the other is downscaled. In each output channel, the ISP provides both, the streaming and the AXI interface. This section describes only the streaming output protocol.

10.5.2.1 Output signal description table

The following table shows the full resolution and downscaled image video output signals. The signal properties are similar for both the interfaces.

Signal name	Direction	Dimension	Description
`<fr/ds>_hvalid_out`	Out	1	'Horizontal valid' signal: '1' means image data is present; '0' means blanking between lines and between frames
`<fr/ds>_vvalid_out`	Out	1	'Vertical valid' signal: '1' means frame is active; '0' means vertical blanking between frames.
`<fr/ds>_uvvalid_out`	Out	1	UV subsampled data qualifier, “1” when U and/or V pixel data is valid. This has no meaning for RGB formats.
`<fr/ds>_vdata_r_y_out`	Out	[15:0]	Pixel data, Red or Y channel. Valid data is 10-bit MSB aligned. The 16bit data is sent out so that some of the internal data can be monitored in the bypass mode.
`<fr/ds>_vdata_g_u_out`	Out	[15:0]	Pixel data, Green or Cb or U channel. Valid data is 10-bit MSB aligned. The 16bit data is sent out so that some of the internal data can be monitored in the bypass mode.

Signal name	Direction	Dimension	Description
`<fr/ds>_vdata_b_v_out`	Out	[15:0]	Pixel data, Blue or Cr or V channel. Valid data is 10-bit MSB aligned. The 16bit data is sent out so that some of the internal data can be monitored in the bypass mode.

10.5.2.2 Video monitor output

This interface provides some observability into the internal states of the ISP. Mali-IV009 can be configured to provide certain module's output through this channel.

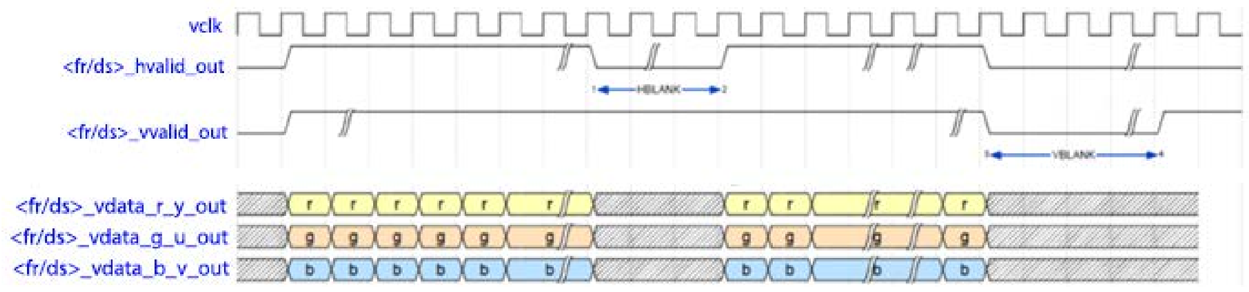
Signal name	Direction	Dimension	Description
`monitor_hvalid_o`	out	1	'Horizontal valid' signal: '1' means image data is present. '0' means blanking between lines and between frames.
`monitor_vvalid_o`	out	1	'Vertical valid' signal: '1' means frame is active. '0' means vertical blanking between frames.
`monitor_vdata_o`	out	[35:0]	Monitor data, MSB aligned.

10.5.2.3 Infrared pixel output interface

This interface provides the interpolated Infrared (Ir) pixel for the RGBIr sensors.

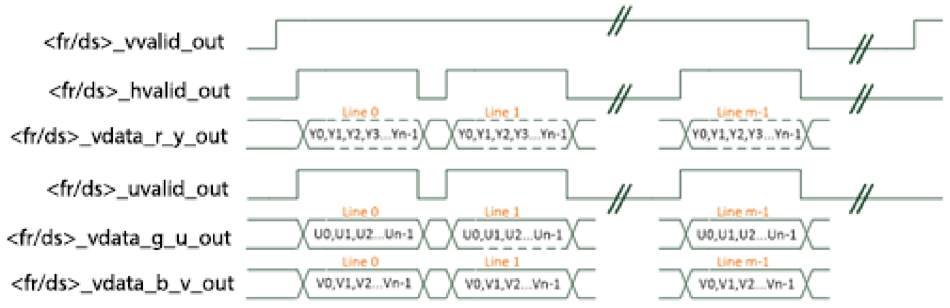
Signal name	Direction	Dimension	Description
`ir_hvalid_o`	out	1	'Horizontal valid' signal: '1' means image data is present. '0' means blanking between lines and between frames.
`ir_vvalid_o`	out	1	'Vertical valid' signal: '1' means frame is active. '0' means vertical blanking between frames.
`ir_vdata_o`	out	[11:0]	Interpolated Ir data, MSB aligned.
`ir_mask_o`	out	1	Valid pulse for the Ir pixel location in the sensor quad.

10.5.2.4 RGB Video Output Timing Diagram

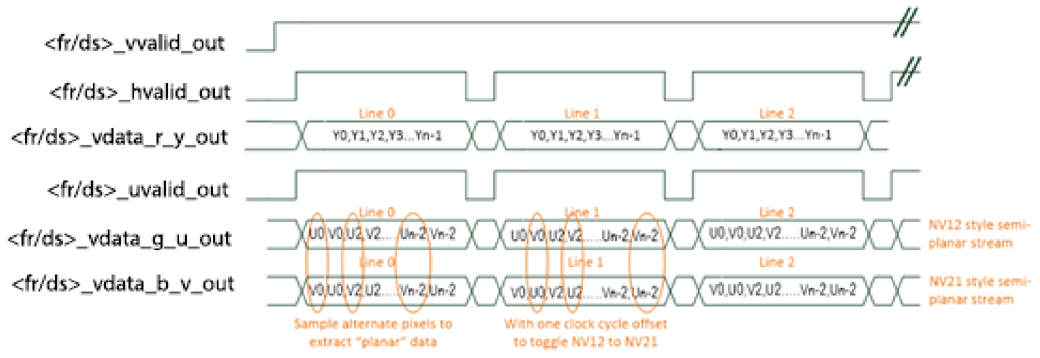


10.5.2.5 Video Output YUV formats

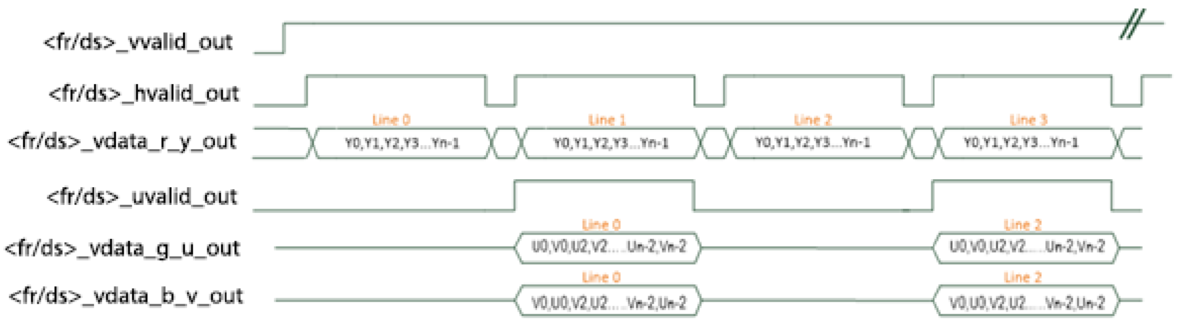
10.5.2.5.1YUV 4:4:4



10.5.2.5.2YUV 4:2:2


<fr/ds>_vvalid_out
<fr/ds>_hvalid_out
<fr/ds>_vdata_r_y_out
Line 0
Line 1
Line 2
Y0,Y1,Y2,Y3...Yn-1
<fr/ds>_uvalid_out
<fr/ds>_vdata_g_u_out
U0,V0,U2,V2.....Un-2,Vn-2
<fr/ds>_vdata_b_v_out
V0,U0,V2,U2.....Vn-2,Un-2
NV12 style semi-planar stream
NV21 style semi-planar stream
Sample alternate pixels to extract "planar" data
With one clock cycle offset to toggle NV12 to NV21

10.5.2.5.3 YUV 4:2:0



10.6 DMA monitor interface

The following signals are OPTIONAL monitor signals. These can be used to improve system performances.

Clock domain: aclk

Signal	Description
`*_words_used`	The count of words used in wfifo.0 to (2 ^ fifo_aw). So, the max value is 128 for fifo_aw= 7 In normal operation the wfifo monitor will show a small fill level during the frame, but during periods of high AXI latency the fill level will increase. The wfifo/rfifo monitors can be used to monitor the active fill level during a frame for simulation/debug purposes. It can also be connected to an external custom fifo monitor circuit to generate priority requests or alarms as needed for specific applications.
`*_ used_valid`	The valid signal for the words_used.
`*_linetick_out`	This is an indicator when the DMA writer sends a programmable number of lines of the video data.

dma_writer_wfifo_monitor

Signal	Direction	Dimension
`fr_dma_writer_wfifo_words_used`	out	[AXI_WFIFO_AW:0]
`fr_dma_writer_wfifo_words_used_valid`	out	1
`fr_dma_writer_linetick_out`	out	1

uv_dma_writer_wfifo_monitor

Signal	Direction	Dimension
`fr_uv_dma_writer_wfifo_words_used`	out	[AXI_WFIFO_UV_AW:0]
`fr_uv_dma_writer_wfifo_words_used_valid`	out	1

Signal	Direction	Dimension
`fr_uv_dma_writer_linetick_out`	out	1

dma_writer_wfifo_monitor

Signal	Direction	Dimension
`ds_dma_writer_wfifo_words_used`	out	[AXI_WFIFO_UV_AW:0]
`ds_dma_writer_wfifo_words_used_valid`	out	1
`ds_dma_writer_linetick_out`	out	1

uv_dma_writer_wfifo_monitor

Signal	Direction	Dimension
`ds_uv_dma_writer_wfifo_words_used`	out	[AXI_WFIFO_AW:0]
`ds_uv_dma_writer_wfifo_words_used_valid`	out	1
`ds_uv_dma_writer_linetick_out`	out	1

temper_fb_<msb/lsb>_wfifo_monitor

Signal	Direction	Dimension
`temper_fb_<msb/lsb>_wfifo_words_used`	out	[AXI_WFIFO_AW:0]
`temper_fb_<msb/lsb>_wfifo_words_used_valid`	out	1

temper_fb_<msb/lsb>_rfifo_monitor

Signal	Direction	Dimension
`temper_fb_<msb/lsb>_rfifo_words_used`	out	[AXI_RFIFO_AW:0]
`temper_fb_<msb/lsb>_rfifo_words_used_valid`	out	1

10.7 Scan interface

Clock domain: asynchronous

This is the DFT test control or scan enable signal for the manual clock gating cells used inside ISP. This is connected to the test enable signal of the clock gate cell. This must be connected to the appropriate test pin of the SoC.

Signal	Direction	Dimension
`scantest_enable`	in	1

10.8 Interrupt interface

Clock domain: vclk

This is the combined interrupt signal.

Signal	Direction	Dimension
`interrupt`	out	1

10.9 Miscellaneous interface

Clock domain: vclk

Signal	Direction	Dimension
`isp_multictx_frame_req`	out	1

In the multi-context use-case, this signal indicates that the ISP is ready to accept a new frame. When the new frame starts coming, this signal goes low. This signal goes high only when it is in the multi-context mode, the configuration for the upcoming frame is done and the current frame is completed.

Note: This signal is optional.

The `max_addr_dly_<fr/ds>` signals are to limit the maximum allowed address location for the delay line memories.

Signal	Direction	Dimension
`max_addr_dly_line_fr`	in	[15:0]
`max_addr_dly_line_ds`	in	[15:0]

11 Clock resets integration

This section describes the clock reset integration options available with the Mali-IV009 ISP.

11.1 Clock and reset

The Mali-IV009 has the following clock and reset domains:

- vclk-rstn
- aclk-aresetn

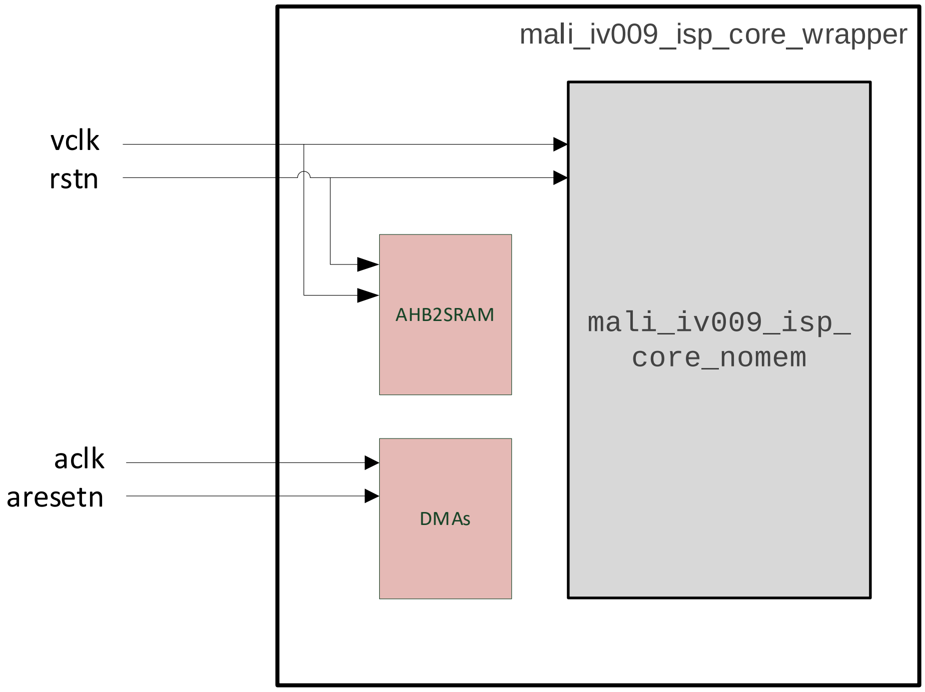


Figure 6. ISP clock and reset domain

All the resets are asynchronous and their assertion is asynchronous. However, the de-assertion **must be** synchronous to the vclk and aclk respectively. Refer to Figure 7.

The ISP does not require any particular reset sequence for the occurrence of aresetn and rstn. They can occur in any order. But the following conditions must be satisfied:

- All the reset sequencing must be completed before any ISP configuration starts.
- Both the resets must be active together.

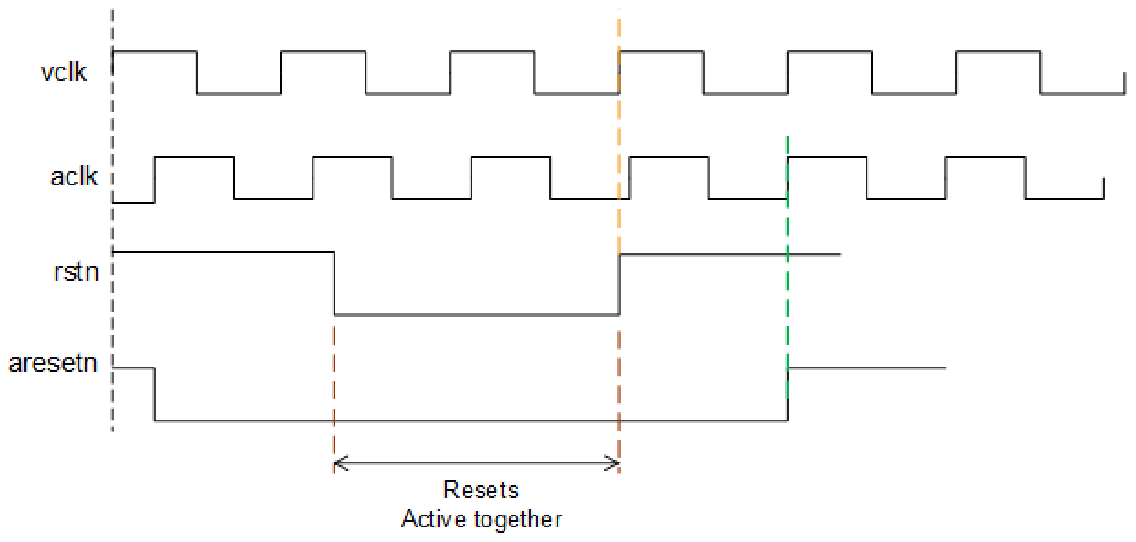


Figure 7. Clock and reset timing diagram

Note: ***It is recommended that the SoC must have an external reset control mechanism such that the system level soft reset can be driven to the ISP primary rstn and aresetn pins also. This can be very useful if the system detects error or hung condition and the ISP needs to be reset separately.***

11.2 Stalling the video pipeline processing

The clock enable signal (`vcke`) can be used to stall the video pipeline processing. Any AXI domain processing is unaffected by the `vcke`. During the time that the clock enable signal is de-asserted the IP does not accept input video data or move data down the video pipeline. It maintains the current state of the video pipeline.

When you use `vcke` to stall the IP processing you must consider the following requirements related to stalled horizontal blanking and flushing of the pipeline:

- Stalled horizontal blanking: At least 32 cycles (pixel clock cycles) of the horizontal blanking which is not stalled is required. See Figure 8.
- Flushing of the pipeline: When the end of a frame occurs at the start of the pipeline there are still many lines of unprocessed video data within the pipeline. The `vcke` must be asserted to move this data down the pipeline and ‘Flush’ it, so cycles where `vcke` is de-asserted will stall the flushing of the pipeline. The pipeline will have completed flushing when the ISP end interrupt is asserted.

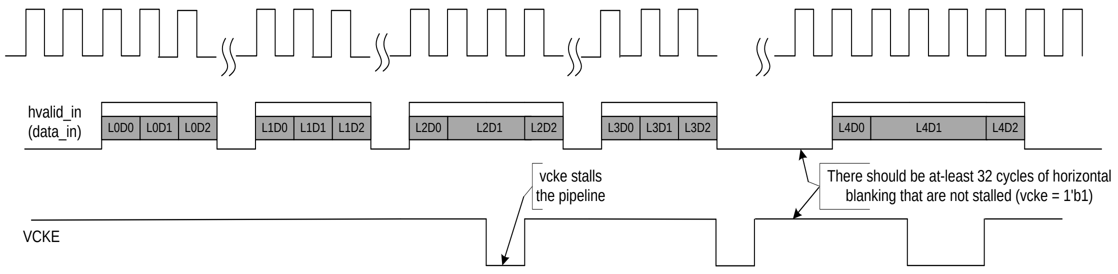


Figure 8. Interface Requirements with Stalled Pipeline Functionality

12 Running the simulation

This section describes how to run a test case and compare results using VCS (Synopsys) and IUS (Cadence).

The Out of Box testbench is located in the following location:
`mali_iv009/logical/testbench/execution_tb/isp_testbench_wrapper.v`

Note:* *Please note that the execution test bench is not the development test bench. This is only Out of Box test bench whose scope is limited ONLY to run the test cases provided and only with the options provided in this section.

The out of box execution test-bench has the following limitations.

- The VCKE signal is always forced to 1.
- AXI slave interfaces are always ready.
- Only both the settings of ARM_ISP_LITE_ENABLE parameter can be verified in this test bench.
- Other parameters must be kept in their default values.

The test cases are located in the following location:
`mali_iv009/logical/testbench/tests/`

The following table provides a short description of the tests. All the test cases can be run in both the settings of the following parameters:

Test Name	Test Description
`isp_linear_single_context_test`	• This test has 2 frames. • Each frame has 480x240 resolution. • The input data is 12-bit raw mode. • The Cfa-pattern is set to RGGB mode. • Static defect pixel and the 3D LUT are bypassed. The temper is configured in temper-2 mode.
`isp_dol_single_context_test`	• This test has 2 frames. • Each frame has 480x240 resolution. • The sensor data is configured in 3-exposure

Test Name	Test Description
	DOL mode. • The shading is only the mesh shading and temper is configured in temper-3 mode. • Static defect pixel and the 3D LUT are bypassed.
`isp_single_context_ping_only_test`	In this test, only the PING address space is used for 4 frames. It is programmed only once. The ping pong select is set to Software control mode and always set to PING mode.
`isp_2ctx_linear_mode_test`	• Input resolution: 480x240. 4 frames. • This test has 2 contexts. • Each context is in linear mode. • FR DMA output is in RGB565 mode. Streaming output is RGB data. • Temper is configured in Temper-3 mode. Sinter, CAC, purple fringing correction are enabled. Only Mesh shading is enabled, radial shading is bypassed. 5-bit AEXP, 1024-bin AEXP, AF, AWB are enabled. 3D LUT bypassed. • DS scaler is bypassed. • DS data is cropped to 128x128. • DS DMA is in RGB565 mode.
`isp_4ctx_mixed_mode_test`	• Input resolution: 128x128. 8 frames. • This test has 4 contexts. • 3 contexts are in linear mode. • 1 context is in 4-exp DOL mode. • FR DMA output is in NV12 mode. Streaming output is YUV420 data. • DS DMA is in RGB565 mode. • Scaler is bypassed for the DOL context. • Scaler is scaled to 64x64 for the other 3 contexts. • Context-1 uses scaler coeff sets [2,3]. • Context-2 uses scaler coeff sets [4,5].

Test Name	Test Description
	• Context-3 uses scaler coeff sets [6,7].

The common Makefile is in the following location:

```
logical/testbench/execution_tb/run
```

The make targets are as follows:

Makefile Target	Description
`help`	'make help' shows information about the targets (default)
`all`	Performs clean, compile and then runs the test
`compile`	compiles the design or testbench
`run`	executes the simulation for specific testcases
`clean`	removes the simulation and compile directories

The command line option to run the Makefile is as follows:

```
make <compile/run/all> SIM=<IUS/VCS> TESTNAME=<test case name>
ISP_LITE=<0/1>
```

Where:

`IUS/VCS`	: The simulator version for IUS or VCS
`TSTNAME`	: Specifies the name of the test-case
`ISP_LITE`	: This enables and disables the ISP_LITE feature.

The ISP_LITE option should be used only when the `ARM_ISP_LITE_ENABLE` parameter is set to 1 in the `logical/mali_iv009_core/config/mali_iv009_top_global_param.vi` file. The parameter value in the configuration file and the runtime option must always match.

The out-of-box (OOB) testbench uses python scripts for format conversion of the dumps and compare the RTL output against the c-model output. You must already have the following tools installed before running the tests.

- Python 2.7.13
- NumPy package of version 1.11.0 for python 2.7

- GCC version 4.9.2 or later and Libstdc++ 3.4.20 or 3.4.21
- The execution test-bench self-checking feature depends on above mentioned package of NumPy and Python version. **This is verified ONLY using Redhat Linux enterprise 6 version**.

To run a test case.

1. Change the directory to `logical/testbench/execution_tb/run`.
2. Type `make clean`.
3. Type
   `make all TESTNAME=<test_name> SIM=<VCS/IUS> ISP_LITE=<0/1>`

For example, to run the `isp_dol_single_context_test` test in default mode using VCS, type the following command:

```
make all TESTNAME=isp_dol_single_context_test SIM=VCS
```

To run the same test in ISP_LITE mode, change the parameter value of `ARM_ISP_LITE_ENABLE` to 1 and type the following command

```
make all TESTNAME=isp_dol_single_context_test SIM=VCS ISP_LITE=1
```

Note: ***When changing simulator, please perform*** *`make clean`* ***before running a test***

When the test is completed, the report states TEST_PASS or TEST_FAIL. At the end of the test, `diff_log.log` file will be generated. If the test fails, then the `diff_log.log` file will report which intermediate or final output is mismatching.

13 Key implementation points

There are some key points to consider for the implementation of the design.

The following table shows where to find information for the key implementation process tasks.

Implementation process task		Description	See
1	Configure RTL	How to set up the configurable parameters.	Configuring the Mali-IV009
2	Configure architectural clock gating	How to configure architectural clock gating.	Clock gate cell integration
3	Integrate memory macros	How to integrate and modify the memories.	Memory integration
4	Test memory macros integration	How to test memory macro integration.	Memory integration
5	Perform synthesis and architecture scan chains	How to perform synthesis and architecture scan chains (*Design For Test* (DFT)).	Key Implementation Points
6	Create floorplan	Guidelines on how to create floorplans.	Floorplans
7	Perform physical implementation using a reference floorplan	How to perform and verify the physical implementation.	Floorplans
8	Perform logical verification		
9	Perform LVS and DRC checks		
10	Perform timing verification		
11	Perform equivalence checking	How to perform logical equivalence checking.	Equivalence Checking
12	Signoff	How to satisfy all applicable verification criteria before you sign off the design.	Sign-off

Table 16. Key Implementation process tasks

13.1 Other considerations for the implementation

The following schemes must be considered during implementation.

13.1.1 Clocking scheme

The Mali-IV009 ISP uses two asynchronous clocks aclk and vclk with a target frequency of 800Mhz and 600Mhz respectively. In the reference implementation each clock is assigned a different clock group and a virtual clock.

The Inputs and outputs are synchronous to their respective clocks unless specific implementation choices are made.

```
set_clock_groups -asynchronous \

     -group [get_clocks aclk v_aclk] \

     -group [get_clocks vclk v_vclk]
```

13.1.2 Timing exceptions for resets

No multicycle exceptions have been set for the Mali-IV009 ISP.

13.1.3 Timing exceptions

No multicycle exceptions have been set for the Mali-IV009 ISP.

13.1.4 DFT timing constraints

The timing constraints do not contain constraints for DFT or MBIST signals, since they are not instantiated in the RTL and are specific to your own implementation. The reference implementation that was performed included insertion of scan chains plus introduction of a signal to control the scan enable port of the registers and the clock gates.

The signal port to use for this is called `scantest_enable`.

13.1.5 VCKE port placement consideration

When placing the vcke port, you must consider the position of the register driving this port.

The register should be placed as close to the Mali-IV009 ISP vcke port as possible to help with timing inside the Mali-IV009 ISP.

13.1.6 Gray code consideration

Considerations for the gray code registers should be made during CTS for their clock domain crossing with regards to meta stability.

The transfer of data between Q-D of the registers will need to be within 1 clock cycle of the fasted clock, minus the max skew across the source registers plus the max skew across the destination registers.

The skew for each source and destination pair should be kept to a minimum across the bus.

For Example: Max Delay = Fastest clock period - (Source register max skew + Destination registers max skew)

The source and destination registers for each pair should be placed close together with no buffering between the source registers Q pin and destination registers D pin.

This guidance should be applied across each instance of gray code register pairing.

The pairings of which can be found in the following table.

Details	Src/Dest	Register Name Pairs	Clock
DMA Writer			
	Source	mali_iv009_isp_core_nomem_i/inst_isp_nr_cluster/inst_temper_dma_lsb_writer/inst_dma_writer/inst_writer/wrfifo/apical_fifo/wfifo_wr_addr_gray_del_reg[*]/Q	VCLK
	Destination	mali_iv009_isp_core_nomem_i/inst_isp_nr_cluster/inst_temper_dma_lsb_writer/inst_dma_writer/inst_writer/wrfifo/apical_fifo/rfifo_wr_addr_gray_r1_reg[*]/D	ACLK
	Source	mali_iv009_isp_core_nomem_i/inst_isp_nr_cluster/inst_temper_dma_msb_writer/inst_dma_writer/inst_writer/wrfifo/apical_fifo/wfifo_wr_addr_gray_del_reg[*]/Q	VCLK
	Destination	mali_iv009_isp_core_nomem_i/inst_isp_nr_cluster/inst_temper_dma_msb_writer/inst_dma_writer/inst_writer/wrfifo/apical_fifo/rfifo_wr_addr_gray_r1_reg[*]/D	ACLK
	Source	mali_iv009_isp_core_nomem_i/gen_ds_pile_en.inst_ds_isp_output_cluster/inst_dma_writer/inst_dma_writer/inst_writer/wrfifo/apical_fifo/wfifo_wr_addr_gray_del_reg[*]/Q	VCLK
	Destination	mali_iv009_isp_core_nomem_i/gen_ds_pile_en.inst_ds_isp_output_cluster/inst_dma_writer/inst_dma_writer/inst_writer/wrfifo/apical_fifo/rfifo_wr_addr_gray_r1_reg[*]/D	ACLK
	Source	mali_iv009_isp_core_nomem_i/gen_ds_pile_en.inst_ds_isp_output_cluster/inst_uv_dma_writer/inst_dma_writer/inst_writer/wrfifo/apical_fifo/wfifo_wr_addr_gray_del_reg[*]/Q	VCLK
	Destination	mali_iv009_isp_core_nomem_i/gen_ds_pile_en.inst_ds_isp_output_cluster/inst_uv_dma_writer/inst_dma_writer/inst_writer/wrfifo/apical_fifo/rfifo_wr_addr_gray_r1_reg[*]/D	ACLK
	Source	mali_iv009_isp_core_nomem_i/inst_fr_isp_output_cluster/inst_dma_writer/inst_dma_writer/inst_writer/wrfifo/apical_fifo/wfifo_wr_addr_gray_del_reg[*]/Q	VCLK
	Destination	mali_iv009_isp_core_nomem_i/inst_fr_isp_output_cluster/inst_dma_writer/inst_dm	ACLK

Details	Src/Dest	Register Name Pairs	Clock
		a_writer/inst_writer/wrfifo/apical_fifo/rfifo_wr_addr_gray_r1_reg[*]/D	
	Source	mali_iv009_isp_core_nomem_i/inst_fr_isp_output_cluster/inst_uv_dma_writer/inst_dma_writer/inst_writer/wrfifo/apical_fifo/wfifo_wr_addr_gray_del_reg[*]/Q	VCLK
	Destination	mali_iv009_isp_core_nomem_i/inst_fr_isp_output_cluster/inst_uv_dma_writer/inst_dma_writer/inst_writer/wrfifo/apical_fifo/rfifo_wr_addr_gray_r1_reg[*]/D	ACLK
	Source	mali_iv009_isp_core_nomem_i/inst_isp_nr_cluster/inst_temper_dma_lsb_writer/inst_dma_writer/inst_writer/wrfifo/apical_fifo/rfifo_rd_addr_gray_del_reg[*]/Q	ACLK
	Destination	mali_iv009_isp_core_nomem_i/inst_isp_nr_cluster/inst_temper_dma_lsb_writer/inst_dma_writer/inst_writer/wrfifo/apical_fifo/wfifo_rd_addr_gray_r1_reg[*]/D	VCLK
	Source	mali_iv009_isp_core_nomem_i/inst_isp_nr_cluster/inst_temper_dma_msb_writer/inst_dma_writer/inst_writer/wrfifo/apical_fifo/rfifo_rd_addr_gray_del_reg[*]/Q	ACLK
	Destination	mali_iv009_isp_core_nomem_i/inst_isp_nr_cluster/inst_temper_dma_msb_writer/inst_dma_writer/inst_writer/wrfifo/apical_fifo/wfifo_rd_addr_gray_r1_reg[*]/D	VCLK
	Source	mali_iv009_isp_core_nomem_i/gen_ds_pile_en.inst_ds_isp_output_cluster/inst_dma_writer/inst_dma_writer/inst_writer/wrfifo/apical_fifo/rfifo_rd_addr_gray_del_reg[*]/Q	ACLK
	Destination	mali_iv009_isp_core_nomem_i/gen_ds_pile_en.inst_ds_isp_output_cluster/inst_dma_writer/inst_dma_writer/inst_writer/wrfifo/apical_fifo/wfifo_rd_addr_gray_r1_reg[*]/D	VCLK
	Source	mali_iv009_isp_core_nomem_i/gen_ds_pile_en.inst_ds_isp_output_cluster/inst_uv_dma_writer/inst_dma_writer/inst_writer/wrfifo/apical_fifo/rfifo_rd_addr_gray_del_reg[*]/Q	ACLK
	Destination	mali_iv009_isp_core_nomem_i/gen_ds_pile_en.inst_ds_isp_output_cluster/inst_uv_dma_writer/inst_dma_writer/inst_writer/wrfifo/apical_fifo/wfifo_rd_addr_gray_r1_reg[*]/D	VCLK
	Source	mali_iv009_isp_core_nomem_i/inst_fr_isp_output_cluster/inst_dma_writer/inst_dma_writer/inst_writer/wrfifo/apical_fifo/rfifo_rd_addr_gray_del_reg[*]/Q	ACLK
	Destination	mali_iv009_isp_core_nomem_i/inst_fr_isp_output_cluster/inst_dma_writer/inst_dma_writer/inst_writer/wrfifo/apical_fifo/wfifo_rd_addr_gray_r1_reg[*]/D	VCLK
	Source	mali_iv009_isp_core_nomem_i/inst_fr_isp_output_cluster/inst_uv_dma_writer/inst_dma_writer/inst_writer/wrfifo/apical_fifo/rfifo_rd_addr_gray_del_reg[*]/Q	ACLK

Details	Src/Dest	Register Name Pairs	Clock
	Destination	mali_iv009_isp_core_nomem_i/inst_fr_isp_output_cluster/inst_uv_dma_writer/inst_dma_writer/inst_writer/wrfifo/apical_fifo/wfifo_rd_addr_gray_r1_reg[*]/D	VCLK
DMA reader			
	Source	mali_iv009_isp_core_nomem_i/inst_isp_nr_cluster/inst_temper_dma_lsb_reader/inst_dma_reader/inst_reader/rdfifo/apical_fifo/wfifo_wr_addr_gray_del_reg[*]/Q	ACLK
	Destination	mali_iv009_isp_core_nomem_i/inst_isp_nr_cluster/inst_temper_dma_lsb_reader/inst_dma_reader/inst_reader/rdfifo/apical_fifo/rfifo_wr_addr_gray_r1_reg[*]/D	VCLK
	Source	mali_iv009_isp_core_nomem_i/inst_isp_nr_cluster/inst_temper_dma_msb_reader/inst_dma_reader/inst_reader/rdfifo/apical_fifo/wfifo_wr_addr_gray_del_reg[*]/Q	ACLK
	Destination	mali_iv009_isp_core_nomem_i/inst_isp_nr_cluster/inst_temper_dma_msb_reader/inst_dma_reader/inst_reader/rdfifo/apical_fifo/rfifo_wr_addr_gray_r1_reg[*]/D	VCLK
	Source	mali_iv009_isp_core_nomem_i/inst_isp_nr_cluster/inst_temper_dma_lsb_reader/inst_dma_reader/inst_reader/rdfifo/apical_fifo/rfifo_rd_addr_gray_del_reg[*]/Q	VCLK
	Destination	mali_iv009_isp_core_nomem_i/inst_isp_nr_cluster/inst_temper_dma_lsb_reader/inst_dma_reader/inst_reader/rdfifo/apical_fifo/wfifo_rd_addr_gray_r1_reg[*]/D	ACLK
	Source	mali_iv009_isp_core_nomem_i/inst_isp_nr_cluster/inst_temper_dma_msb_reader/inst_dma_reader/inst_reader/rdfifo/apical_fifo/rfifo_rd_addr_gray_del_reg[*]/Q	VCLK
	Destination	mali_iv009_isp_core_nomem_i/inst_isp_nr_cluster/inst_temper_dma_msb_reader/inst_dma_reader/inst_reader/rdfifo/apical_fifo/wfifo_rd_addr_gray_r1_reg[*]/D	ACLK

14 Floorplans

The Arm Mali-IV009 ISP is a design with a high RAM density.

A good floorplan and placement of the memory macro cells are crucial to maximize performance.

The following figure is a process diagram that shows the top-level inputs, resources, outputs, controls and constraints for floorplanning.

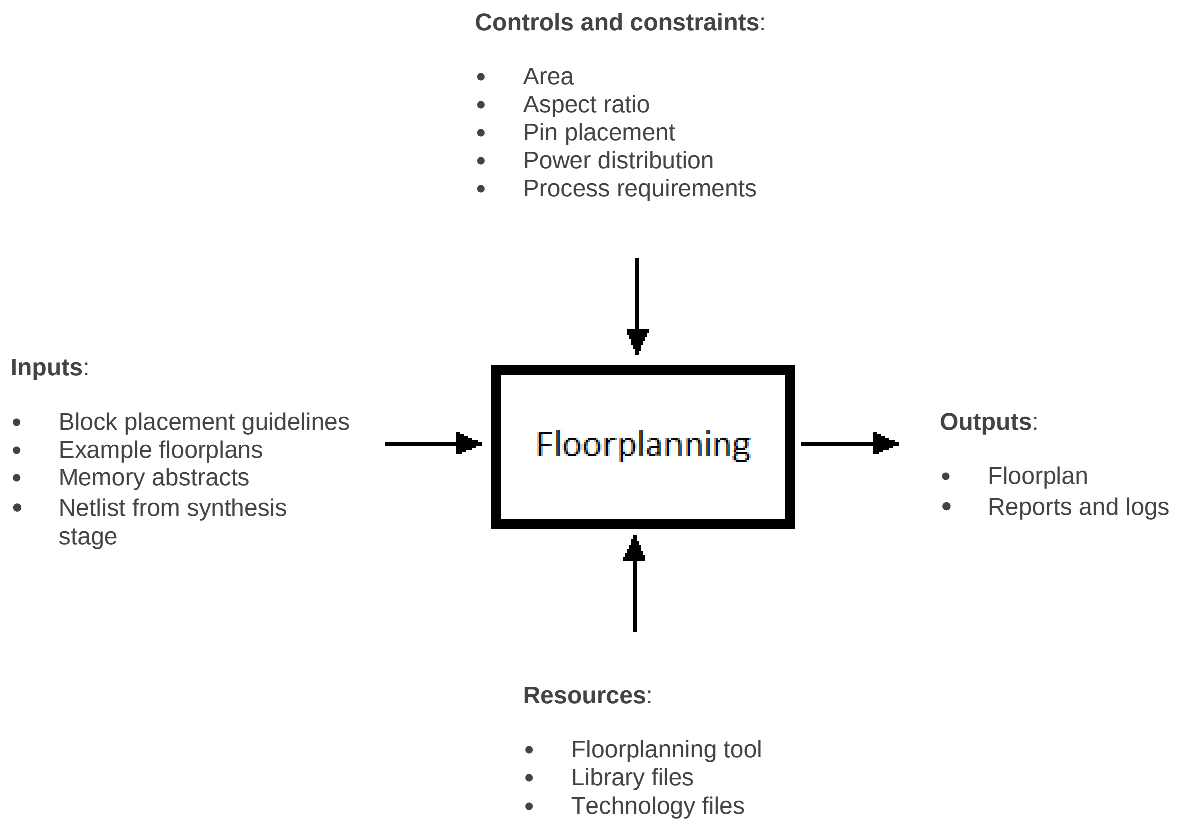


Figure 9. The Floorplan process

14.1 Resource requirements for floorplans

Arm recommends that you have suitable EDA tools and computer resources for floorplanning.

14.2 Controls and constraints for floorplans

There are controls and constraints that can influence floorplanning.

- Pin placement.
- Area.
- Aspect ratio.
- Power distribution.
- Process requirements.

14.3 Inputs for floorplans

Inputs are specific to your floorplanning tool, and can include the following:

- Example floorplans.
- Block placement guideline.
- Pin placement.
- Power distribution.
- Placement blockages.
- Memory macro cell placement.

14.4 Considerations for floorplans

There are various factors to consider in optimizing your floorplan.

Take care in the placement of the memory cells because they occupy a large area of the floorplan and require significant routing resource. The placement of the memories affects the optimum standard cell placement.

14.4.1 Other floorplan considerations

- Consider creating large memories from banks of smaller memory.
- Use register file memories where applicable.
- Consider using a keepout margin for the memories to avoid routing congestions.
- Consider channels between large banks of memories to avoid timing and max transition violations.

- Use the flylines feature if the floorplanning tool allows it.
- The aspect ratios shown for memories are arbitrary. They might not be achievable using your memory compiler.
- It might be necessary to adjust the floorplan if your technology prevents you from routing over the memory blocks.

14.5 Floorplan examples

This section describes floorplan examples.

The following figure shows the ISP Default floorplan example. It is not to scale.

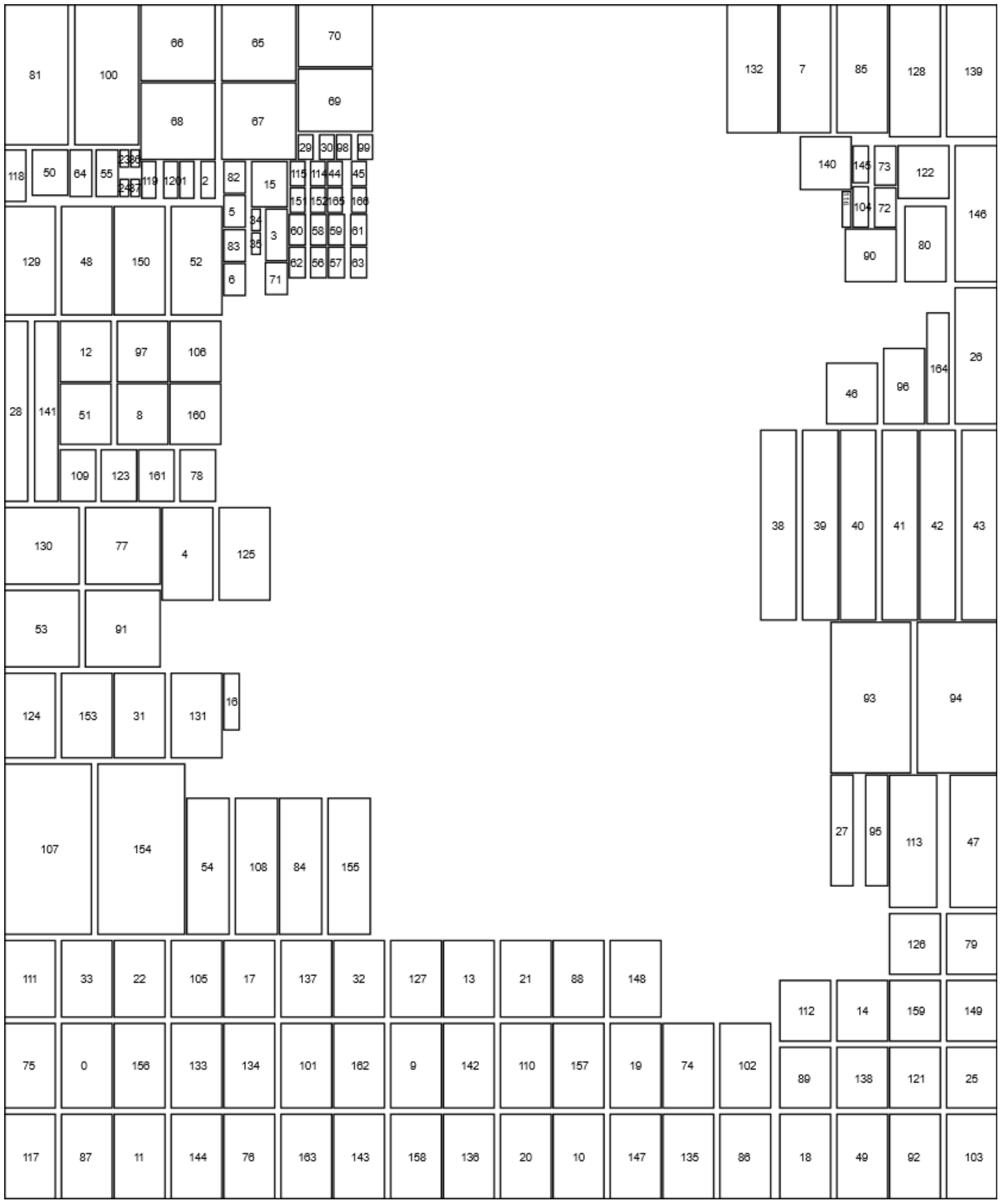


Figure 10. ISP Default floorplan example

The following table shows a key for the ISP Default floorplan example.

Number	Full name
0	mem_top_i/sinter3_rams.inst_mem_sinter3_delay_line_mem_16
1	mem_top_i/inst_mem_decompander1_pong_mem
2	mem_top_i/inst_mem_decompander1_pong_mem2
3	mem_top_i/inst_mem_ihist_stats_mem2
4	mem_top_i/inst_mem_iridix_delay_line_data_mem_1
5	mem_top_i/inst_mem_radial_shading_pong_mem
6	mem_top_i/inst_mem_radial_shading_pong_mem2
7	mem_top_i/inst_mem_uv_mean_rfilter_mem
8	mem_top_i/inst_mem_frame_stitch_dly2_mem_1
9	mem_top_i/sinter3_rams.inst_mem_sinter3_delay_line_mem_05
10	mem_top_i/sinter3_rams.inst_mem_sinter3_delay_line_mem_21
11	mem_top_i/sinter3_rams.inst_mem_sinter3_delay_line_mem_29
12	mem_top_i/inst_mem_frame_stitch_dly1_mem_0
13	mem_top_i/sinter3_rams_cac.inst_mem_ca_correction_delay_line_mem_05
14	mem_top_i/inst_mem_demosaic_dly_mem_1
15	mem_top_i/inst_mem_ihist_stats_mem
16	mem_top_i/inst_mem_af_scratch_mem
17	mem_top_i/sinter3_rams_cac.inst_mem_ca_correction_delay_line_mem_03
18	mem_top_i/sinter3_rams.inst_mem_sinter3_delay_line_mem_15
19	mem_top_i/sinter3_rams.inst_mem_sinter3_delay_line_mem_09
20	mem_top_i/sinter3_rams.inst_mem_sinter3_delay_line_mem_22
21	mem_top_i/sinter3_rams_cac.inst_mem_ca_correction_delay_line_mem_04
22	mem_top_i/sinter3_rams_cac.inst_mem_ca_correction_delay_line_mem_09
23	mem_top_i/inst_mem_decompander0_ping_mem
24	mem_top_i/inst_mem_decompander0_ping_mem2
25	mem_top_i/inst_mem_demosaic_dly_mem_7
26	mem_top_i/inst_mem_ds_dma_writer_wfifo_mem
27	mem_top_i/inst_mem_ds_sharpen_dly2_mem_1
28	mem_top_i/inst_mem_frame_stitch_lc2_mem
29	mem_top_i/inst_mem_iridix_fp2_pong_mem
30	mem_top_i/inst_mem_iridix_fp2_pong_mem2
31	mem_top_i/inst_mem_rfe_dly_mem_0
32	mem_top_i/sinter3_rams_cac.inst_mem_ca_correction_delay_line_mem_07
33	mem_top_i/sinter3_rams_cac.inst_mem_ca_correction_delay_line_mem_10
34	mem_top_i/sinter3_rams_cac.inst_mem_ca_correction_filter_ping_mem

Number	Full name
35	mem_top_i/sinter3_rams_cac.inst_mem_ca_correction_filter_pong_mem
36	mem_top_i/inst_mem_decompander0_pong_mem
37	mem_top_i/inst_mem_decompander0_pong_mem2
38	mem_top_i/inst_mem_ds_scaler_vfilt_linemem1_0
39	mem_top_i/inst_mem_ds_scaler_vfilt_linemem1_1
40	mem_top_i/inst_mem_ds_scaler_vfilt_linemem1_2
41	mem_top_i/inst_mem_ds_scaler_vfilt_linemem1_3
42	mem_top_i/inst_mem_ds_scaler_vfilt_linemem1_4
43	mem_top_i/inst_mem_ds_scaler_vfilt_linemem1_5
44	mem_top_i/inst_mem_iridix_fp2_ping_mem
45	mem_top_i/inst_mem_iridix_fp2_ping_mem2
46	mem_top_i/inst_mem_mirror_mem
47	mem_top_i/inst_mem_ds_sharpen_dly1_mem_1
48	mem_top_i/inst_mem_frame_stitch_dly3_mem_1
49	mem_top_i/sinter3_rams.inst_mem_sinter3_delay_line_mem_11
50	mem_top_i/inst_mem_aexp_hist_stats_mem
51	mem_top_i/inst_mem_frame_stitch_dly2_mem_0
52	mem_top_i/inst_mem_frame_stitch_dly3_mem_3
53	mem_top_i/inst_mem_iridix_mem2_3
54	mem_top_i/inst_mem_temper_fb_msb_wfifo_mem
55	mem_top_i/inst_mem_ds_scaler_hfilt_coefmem
56	mem_top_i/sinter3_rams_lut.inst_mem_lut3d_mem_0
57	mem_top_i/sinter3_rams_lut.inst_mem_lut3d_mem_1
58	mem_top_i/sinter3_rams_lut.inst_mem_lut3d_mem_2
59	mem_top_i/sinter3_rams_lut.inst_mem_lut3d_mem_3
60	mem_top_i/sinter3_rams_lut.inst_mem_lut3d_mem_4
61	mem_top_i/sinter3_rams_lut.inst_mem_lut3d_mem_5
62	mem_top_i/sinter3_rams_lut.inst_mem_lut3d_mem_6
63	mem_top_i/sinter3_rams_lut.inst_mem_lut3d_mem_7
64	mem_top_i/inst_mem_ds_scaler_vfilt_coefmem
65	mem_top_i/inst_mem_mesh_shading_pong_mem
66	mem_top_i/inst_mem_mesh_shading_ping_mem
67	mem_top_i/sinter3_rams_cac.inst_mem_ca_correction_mesh_pong_mem
68	mem_top_i/sinter3_rams_cac.inst_mem_ca_correction_mesh_ping_mem
69	mem_top_i/inst_mem_dpc_pong_mem
70	mem_top_i/inst_mem_dpc_ping_mem
71	mem_top_i/inst_mem_cmd_queues

Number	Full name
72	mem_top_i/inst_mem_lumvar_stats_pong_mem
73	mem_top_i/inst_mem_lumvar_stats_ping_mem
74	mem_top_i/sinter3_rams.inst_mem_sinter3_delay_line_mem_10
75	mem_top_i/sinter3_rams.inst_mem_sinter3_delay_line_mem_17
76	mem_top_i/sinter3_rams.inst_mem_sinter3_delay_line_mem_27
77	mem_top_i/inst_mem_iridix_mem2_0
78	mem_top_i/inst_mem_iridix_mem_0
79	mem_top_i/inst_mem_af_delay_line_mem_0
80	mem_top_i/inst_mem_fr_uv_dma_writer_wfifo_mem
81	mem_top_i/inst_mem_metering_stats_pong_mem
82	mem_top_i/inst_mem_radial_shading_ping_mem
83	mem_top_i/inst_mem_radial_shading_ping_mem2
84	mem_top_i/inst_mem_temper_fb_lsb_rfifo_mem
85	mem_top_i/inst_mem_uv_var_rfilter_mem
86	mem_top_i/sinter3_rams.inst_mem_sinter3_delay_line_mem_18
87	mem_top_i/sinter3_rams.inst_mem_sinter3_delay_line_mem_30
88	mem_top_i/sinter3_rams_cac.inst_mem_ca_correction_delay_line_mem_01
89	mem_top_i/inst_mem_demosaic_dly_mem_4
90	mem_top_i/inst_mem_fr_sharpen_dly2_mem_1
91	mem_top_i/inst_mem_iridix_mem2_2
92	mem_top_i/sinter3_rams.inst_mem_sinter3_delay_line_mem_12
93	mem_top_i/inst_mem_ds_scaler_lb_fifo_mem1_0
94	mem_top_i/inst_mem_ds_scaler_lb_fifo_mem1_1
95	mem_top_i/inst_mem_ds_sharpen_dly2_mem_0
96	mem_top_i/inst_mem_ds_uv_dma_writer_wfifo_mem
97	mem_top_i/inst_mem_frame_stitch_dly1_mem_1
98	mem_top_i/inst_mem_iridix_rp_ping_mem
99	mem_top_i/inst_mem_iridix_rp_ping_mem2
100	mem_top_i/inst_mem_metering_stats_ping_mem
101	mem_top_i/sinter3_rams.inst_mem_sinter3_delay_line_mem_03
102	mem_top_i/sinter3_rams.inst_mem_sinter3_delay_line_mem_13
103	mem_top_i/sinter3_rams.inst_mem_sinter3_delay_line_mem_14
104	mem_top_i/inst_mem_aexp_scratch_mem
105	mem_top_i/sinter3_rams_cac.inst_mem_ca_correction_delay_line_mem_08
106	mem_top_i/inst_mem_frame_stitch_dly1_mem_2
107	mem_top_i/inst_mem_temper_delay_line_mem_1
108	mem_top_i/inst_mem_temper_fb_msb_rfifo_mem

Number	Full name
109	mem_top_i/inst_mem_iridix_mem_3
110	mem_top_i/sinter3_rams.inst_mem_sinter3_delay_line_mem_07
111	mem_top_i/sinter3_rams_cac.inst_mem_ca_correction_delay_line_mem_11
112	mem_top_i/inst_mem_demosaic_dly_mem_0
113	mem_top_i/inst_mem_ds_sharpen_dly1_mem_0
114	mem_top_i/inst_mem_iridix_fp1_pong_mem
115	mem_top_i/inst_mem_iridix_fp1_pong_mem2
116	mem_top_i/inst_mem_lumvar_scratch_mem
117	mem_top_i/sinter3_rams.inst_mem_sinter3_delay_line_mem_31
118	mem_top_i/inst_mem_aexp_hist_stats_mem2
119	mem_top_i/inst_mem_decompander1_ping_mem
120	mem_top_i/inst_mem_decompander1_ping_mem2
121	mem_top_i/inst_mem_demosaic_dly_mem_6
122	mem_top_i/inst_mem_fr_cs_conv_dly_mem
123	mem_top_i/inst_mem_iridix_mem_1
124	mem_top_i/inst_mem_rfe_dly_mem_1
125	mem_top_i/inst_mem_iridix_delay_line_data_mem_0
126	mem_top_i/inst_mem_af_delay_line_mem_1
127	mem_top_i/sinter3_rams_cac.inst_mem_ca_correction_delay_line_mem_06
128	mem_top_i/inst_mem_fr_sharpen_dly1_mem_0
129	mem_top_i/inst_mem_frame_stitch_dly3_mem_0
130	mem_top_i/inst_mem_iridix_mem2_1
131	mem_top_i/inst_mem_rfe_dly_mem_3
132	mem_top_i/inst_mem_seg_mean_rfilter_mem
133	mem_top_i/sinter3_rams.inst_mem_sinter3_delay_line_mem_01
134	mem_top_i/sinter3_rams.inst_mem_sinter3_delay_line_mem_02
135	mem_top_i/sinter3_rams.inst_mem_sinter3_delay_line_mem_19
136	mem_top_i/sinter3_rams.inst_mem_sinter3_delay_line_mem_23
137	mem_top_i/sinter3_rams_cac.inst_mem_ca_correction_delay_line_mem_02
138	mem_top_i/inst_mem_demosaic_dly_mem_5
139	mem_top_i/inst_mem_fr_sharpen_dly1_mem_1
140	mem_top_i/inst_mem_fr_sharpen_dly2_mem_0
141	mem_top_i/inst_mem_frame_stitch_lc1_mem
142	mem_top_i/sinter3_rams.inst_mem_sinter3_delay_line_mem_06
143	mem_top_i/sinter3_rams.inst_mem_sinter3_delay_line_mem_25
144	mem_top_i/sinter3_rams.inst_mem_sinter3_delay_line_mem_28
145	mem_top_i/inst_mem_awb_scratch_mem

Number	Full name
146	mem_top_i/inst_mem_fr_dma_writer_wfifo_mem
147	mem_top_i/sinter3_rams.inst_mem_sinter3_delay_line_mem_20
148	mem_top_i/sinter3_rams_cac.inst_mem_ca_correction_delay_line_mem_00
149	mem_top_i/inst_mem_demosaic_dly_mem_3
150	mem_top_i/inst_mem_frame_stitch_dly3_mem_2
151	mem_top_i/inst_mem_iridix_fp1_ping_mem
152	mem_top_i/inst_mem_iridix_fp1_ping_mem2
153	mem_top_i/inst_mem_rfe_dly_mem_2
154	mem_top_i/inst_mem_temper_delay_line_mem_0
155	mem_top_i/inst_mem_temper_fb_lsb_wfifo_mem
156	mem_top_i/sinter3_rams.inst_mem_sinter3_delay_line_mem_00
157	mem_top_i/sinter3_rams.inst_mem_sinter3_delay_line_mem_08
158	mem_top_i/sinter3_rams.inst_mem_sinter3_delay_line_mem_24
159	mem_top_i/inst_mem_demosaic_dly_mem_2
160	mem_top_i/inst_mem_frame_stitch_dly2_mem_2
161	mem_top_i/inst_mem_iridix_mem_2
162	mem_top_i/sinter3_rams.inst_mem_sinter3_delay_line_mem_04
163	mem_top_i/sinter3_rams.inst_mem_sinter3_delay_line_mem_26
164	mem_top_i/inst_mem_ds_cs_conv_dly_mem
165	mem_top_i/inst_mem_iridix_rp_pong_mem
166	mem_top_i/inst_mem_iridix_rp_pong_mem2

Table 17. ISP Default floorplan example blocks

The following figure shows the ISP LITE floorplan example. It is not to scale.

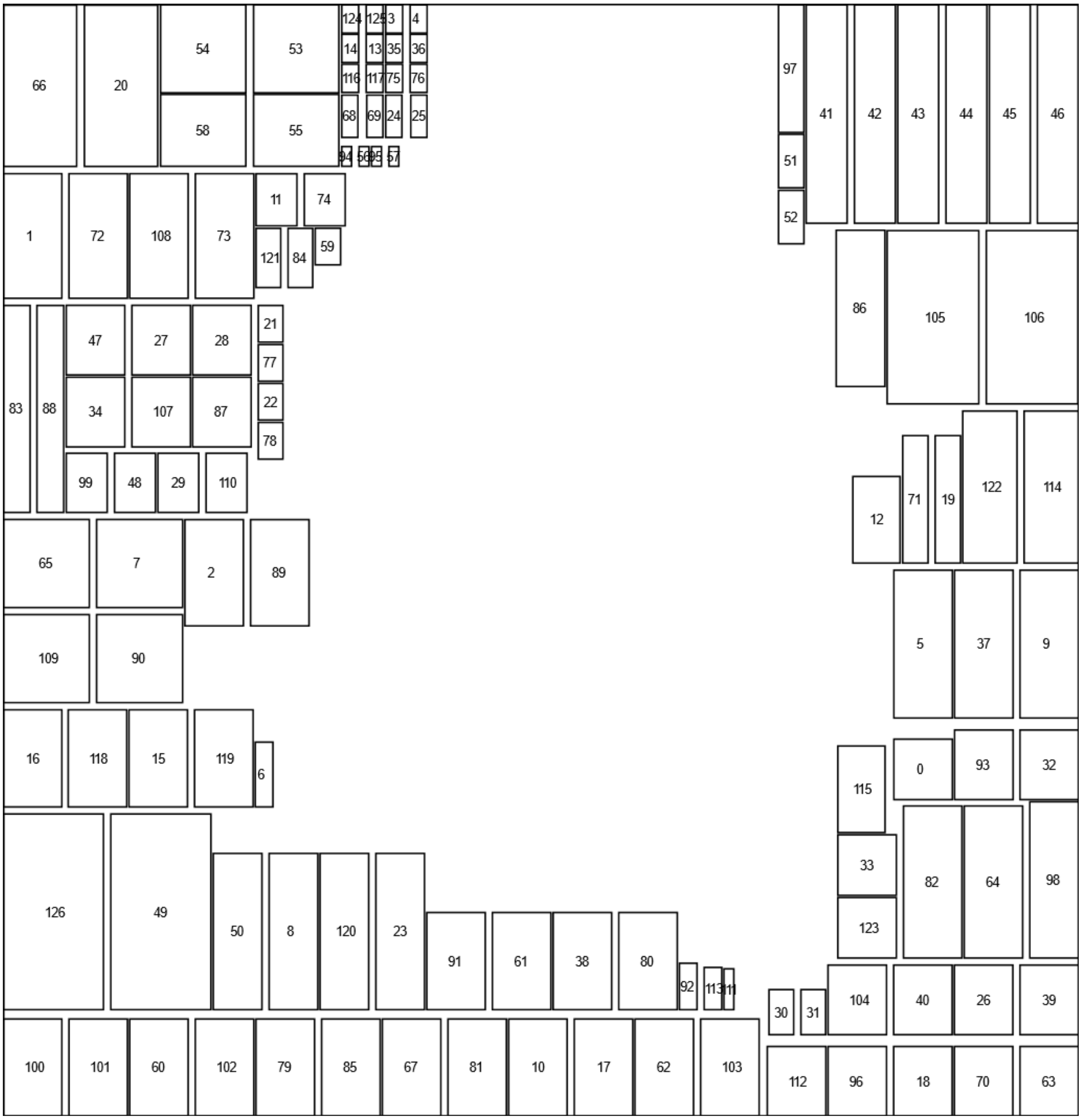


Figure 11. ISP LITE floorplan example

The following table shows a key for the ISP LITE floorplan example.

Number	Full name
0	mem_top_i/inst_mem_fr_cs_conv_dly_mem
1	mem_top_i/inst_mem_frame_stitch_dly3_mem_0
2	mem_top_i/inst_mem_iridix_delay_line_data_mem_1
3	mem_top_i/inst_mem_iridix_rp_ping_mem
4	mem_top_i/inst_mem_iridix_rp_ping_mem2
5	mem_top_i/inst_mem_seg_mean_rfilter_mem
6	mem_top_i/inst_mem_af_scratch_mem
7	mem_top_i/inst_mem_iridix_mem2_0
8	mem_top_i/inst_mem_temper_fb_msb_rfifo_mem
9	mem_top_i/inst_mem_uv_var_rfilter_mem
10	mem_top_i/sinter2p5_rams.inst_mem_sinter2p5_delay_line_mem_12
11	mem_top_i/inst_mem_aexp_hist_stats_mem
12	mem_top_i/inst_mem_ds_uv_dma_writer_wfifo_mem
13	mem_top_i/inst_mem_iridix_fp1_pong_mem
14	mem_top_i/inst_mem_iridix_fp1_pong_mem2
15	mem_top_i/inst_mem_rfe_dly_mem_0
16	mem_top_i/inst_mem_rfe_dly_mem_1
17	mem_top_i/sinter2p5_rams.inst_mem_sinter2p5_delay_line_mem_13
18	mem_top_i/inst_mem_demosaic_dly_mem_5
19	mem_top_i/inst_mem_ds_sharpen_dly2_mem_0
20	mem_top_i/inst_mem_metering_stats_ping_mem
21	mem_top_i/inst_mem_radial_shading_ping_mem
22	mem_top_i/inst_mem_radial_shading_ping_mem2
23	mem_top_i/inst_mem_temper_fb_lsb_wfifo_mem
24	mem_top_i/inst_mem_decompander1_pong_mem
25	mem_top_i/inst_mem_decompander1_pong_mem2
26	mem_top_i/inst_mem_demosaic_dly_mem_2
27	mem_top_i/inst_mem_frame_stitch_dly1_mem_1
28	mem_top_i/inst_mem_frame_stitch_dly1_mem_2
29	mem_top_i/inst_mem_iridix_mem_2
30	mem_top_i/inst_mem_lumvar_stats_ping_mem
31	mem_top_i/inst_mem_lumvar_stats_pong_mem
32	mem_top_i/inst_mem_af_delay_line_mem_0
33	mem_top_i/inst_mem_fr_sharpen_dly2_mem_1
34	mem_top_i/inst_mem_frame_stitch_dly2_mem_0

Number	Full name
35	mem_top_i/inst_mem_iridix_fp2_ping_mem
36	mem_top_i/inst_mem_iridix_fp2_ping_mem2
37	mem_top_i/inst_mem_uv_mean_rfilter_mem
38	mem_top_i/sinter2p5_rams.inst_mem_sinter2p5_delay_line_mem_09
39	mem_top_i/inst_mem_demosaic_dly_mem_3
40	mem_top_i/inst_mem_demosaic_dly_mem_1
41	mem_top_i/inst_mem_ds_scaler_vfilt_linemem1_0
42	mem_top_i/inst_mem_ds_scaler_vfilt_linemem1_1
43	mem_top_i/inst_mem_ds_scaler_vfilt_linemem1_2
44	mem_top_i/inst_mem_ds_scaler_vfilt_linemem1_3
45	mem_top_i/inst_mem_ds_scaler_vfilt_linemem1_4
46	mem_top_i/inst_mem_ds_scaler_vfilt_linemem1_5
47	mem_top_i/inst_mem_frame_stitch_dly1_mem_0
48	mem_top_i/inst_mem_iridix_mem_1
49	mem_top_i/inst_mem_temper_delay_line_mem_0
50	mem_top_i/inst_mem_temper_fb_msb_wfifo_mem
51	mem_top_i/inst_mem_ds_scaler_hfilt_coefmem
52	mem_top_i/inst_mem_ds_scaler_vfilt_coefmem
53	mem_top_i/inst_mem_mesh_shading_pong_mem
54	mem_top_i/inst_mem_mesh_shading_ping_mem
55	mem_top_i/inst_mem_dpc_pong_mem
56	mem_top_i/inst_mem_decompander0_pong_mem
57	mem_top_i/inst_mem_decompander0_pong_mem2
58	mem_top_i/inst_mem_dpc_ping_mem
59	mem_top_i/inst_mem_cmd_queues
60	mem_top_i/sinter2p5_rams.inst_mem_sinter2p5_delay_line_mem_02
61	mem_top_i/sinter2p5_rams.inst_mem_sinter2p5_delay_line_mem_08
62	mem_top_i/sinter2p5_rams.inst_mem_sinter2p5_delay_line_mem_15
63	mem_top_i/inst_mem_demosaic_dly_mem_7
64	mem_top_i/inst_mem_fr_sharpen_dly1_mem_1
65	mem_top_i/inst_mem_iridix_mem2_1
66	mem_top_i/inst_mem_metering_stats_pong_mem
67	mem_top_i/sinter2p5_rams.inst_mem_sinter2p5_delay_line_mem_06
68	mem_top_i/inst_mem_decompander1_ping_mem
69	mem_top_i/inst_mem_decompander1_ping_mem2
70	mem_top_i/inst_mem_demosaic_dly_mem_6
71	mem_top_i/inst_mem_ds_sharpen_dly2_mem_1

Number	Full name
72	mem_top_i/inst_mem_frame_stitch_dly3_mem_1
73	mem_top_i/inst_mem_frame_stitch_dly3_mem_3
74	mem_top_i/inst_mem_ihist_stats_mem
75	mem_top_i/inst_mem_iridix_rp_pong_mem
76	mem_top_i/inst_mem_iridix_rp_pong_mem2
77	mem_top_i/inst_mem_radial_shading_pong_mem
78	mem_top_i/inst_mem_radial_shading_pong_mem2
79	mem_top_i/sinter2p5_rams.inst_mem_sinter2p5_delay_line_mem_04
80	mem_top_i/sinter2p5_rams.inst_mem_sinter2p5_delay_line_mem_10
81	mem_top_i/sinter2p5_rams.inst_mem_sinter2p5_delay_line_mem_11
82	mem_top_i/inst_mem_fr_sharpen_dly1_mem_0
83	mem_top_i/inst_mem_frame_stitch_lc2_mem
84	mem_top_i/inst_mem_ihist_stats_mem2
85	mem_top_i/sinter2p5_rams.inst_mem_sinter2p5_delay_line_mem_05
86	mem_top_i/inst_mem_ds_dma_writer_wfifo_mem
87	mem_top_i/inst_mem_frame_stitch_dly2_mem_2
88	mem_top_i/inst_mem_frame_stitch_lc1_mem
89	mem_top_i/inst_mem_iridix_delay_line_data_mem_0
90	mem_top_i/inst_mem_iridix_mem2_2
91	mem_top_i/sinter2p5_rams.inst_mem_sinter2p5_delay_line_mem_07
92	mem_top_i/inst_mem_aexp_scratch_mem
93	mem_top_i/inst_mem_af_delay_line_mem_1
94	mem_top_i/inst_mem_decompander0_ping_mem
95	mem_top_i/inst_mem_decompander0_ping_mem2
96	mem_top_i/inst_mem_demosaic_dly_mem_4
97	mem_top_i/inst_mem_ds_cs_conv_dly_mem
98	mem_top_i/inst_mem_fr_dma_writer_wfifo_mem
99	mem_top_i/inst_mem_iridix_mem_3
100	mem_top_i/sinter2p5_rams.inst_mem_sinter2p5_delay_line_mem_00
101	mem_top_i/sinter2p5_rams.inst_mem_sinter2p5_delay_line_mem_01
102	mem_top_i/sinter2p5_rams.inst_mem_sinter2p5_delay_line_mem_03
103	mem_top_i/sinter2p5_rams.inst_mem_sinter2p5_delay_line_mem_14
104	mem_top_i/inst_mem_demosaic_dly_mem_0
105	mem_top_i/inst_mem_ds_scaler_lb_fifo_mem1_0
106	mem_top_i/inst_mem_ds_scaler_lb_fifo_mem1_1
107	mem_top_i/inst_mem_frame_stitch_dly2_mem_1
108	mem_top_i/inst_mem_frame_stitch_dly3_mem_2

Number	Full name
109	mem_top_i/inst_mem_iridix_mem2_3
110	mem_top_i/inst_mem_iridix_mem_0
111	mem_top_i/inst_mem_lumvar_scratch_mem
112	mem_top_i/inst_mem_mirror_mem
113	mem_top_i/inst_mem_awb_scratch_mem
114	mem_top_i/inst_mem_ds_sharpen_dly1_mem_1
115	mem_top_i/inst_mem_fr_uv_dma_writer_wfifo_mem
116	mem_top_i/inst_mem_iridix_fp1_ping_mem
117	mem_top_i/inst_mem_iridix_fp1_ping_mem2
118	mem_top_i/inst_mem_rfe_dly_mem_2
119	mem_top_i/inst_mem_rfe_dly_mem_3
120	mem_top_i/inst_mem_temper_fb_lsb_rfifo_mem
121	mem_top_i/inst_mem_aexp_hist_stats_mem2
122	mem_top_i/inst_mem_ds_sharpen_dly1_mem_0
123	mem_top_i/inst_mem_fr_sharpen_dly2_mem_0
124	mem_top_i/inst_mem_iridix_fp2_pong_mem
125	mem_top_i/inst_mem_iridix_fp2_pong_mem2
126	mem_top_i/inst_mem_temper_delay_line_mem_1

Table 18. ISP LITE floorplan example blocks

14.6 Outputs from floorplans

Output files are specific to your floorplanning flow.

They might include:

- Floorplan for replay.
- Fixed pin locations.
- Route guides or blockages.
- Placement blockages.
- Pin placement.
- Initial cell placement.
- Proprietary format for your synthesis or place and route tool.

14.7 Reference data for floorplans

Reference information for floorplans varies according to your product. You must refer to the reference data that is specific to your product.

14.8 Standard cell placement

No automatic placement of standard cells is done during floorplanning.

14.9 Memory inputs

All signals to the memory blocks are direct from the registers.

The memory blocks that are most critical to the floorplan are:

- Larger memory blocks with a larger address setup.
- Memory blocks that are further away from the standard cell area

14.10 Pin placement

Place the pins according to the optimal considerations for the SoC floorplan.

Usually, clock, scan enable, and reset require more attention than other aspects.

14.11 Power grid design

Incorporate the power grid in the floorplan passed to your synthesis tool, so that the tool has a more accurate representation of the available routing resource.

Design the grid according to the requirements of your library. However, Arm recommends a grid that satisfies an IR drop of 2%, for VDD and VSS combined.

15 Equivalence checking

This chapter describes how to test the implementation using equivalence checking.

15.1 About equivalence checking

Use Logical Equivalence Checking (LEC) tools to verify that the Arm Mali-IV009 ISP netlist matches the reference RTL.

Equivalence checkers use formal mathematical techniques to verify logic functions between two implementations of a design. This approach enables you to determine whether the design functionality is consistent between the two implementations. Equivalence checkers allow you to verify that you are maintaining the functionality of the netlist, by building, mapping, and comparing the design. Run LEC between the reference RTL and final sign-off netlist. This process can be split into the following steps:

1. Compare RTL to the synthesis netlist.
2. Compare the synthesis netlist to the final sign-off netlist
3. Review the constraints that you used.

16 Sign-off

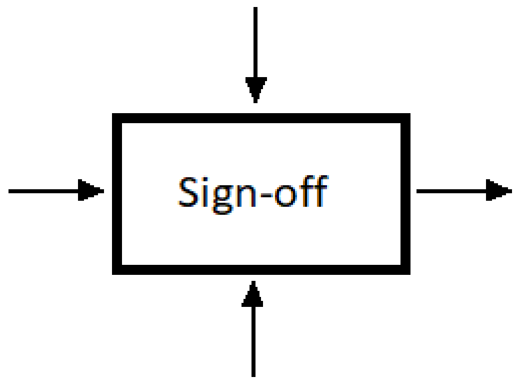
This chapter describes the sign-off criteria. In addition to your normal ASIC flow sign-off checks, there are certain verification criteria to satisfy before you can sign off the Arm Mali-IV009 ISP design.

Controls and constraints:

- Contractual requirements
- Partner sign-off requirements

Inputs:

- Logs and reports



Outputs:

- Signed off macrocell

Resources:

- Signatories

Figure 12. Signoff process

16.1 Obligations for sign-off

Signatories must approve the sign-off of the design in accordance with:

- The terms of the contract with Arm.
- Any other partner sign-off requirements.

See Implementation obligations on page 7 for more information

16.2 Steps for sign-off

To sign off the Arm Mali-IV009 ISP, you must meet the criteria in each stage of the design flow.

16.2.1 RTL integration

Before you begin the synthesis stage, verify the Arm Mali-IV009 ISP deliverables by running the supplied checking test suites on the configured RTL.

Passing the test suites confirms that you have successfully installed the ISP.

16.2.2 Post-synthesis and place-and-route

You must verify the functionality of the final placed-and-routed netlist before you sign off the macro cell.

This verification consists of proving logical equivalence between the validated processor RTL and the final placed-and-routed netlist using formal verification tools.

16.2.3 Post-place-and-route timing

You must use Static Timing Analysis (STA) to verify the timing of the final place-and-routed netlist before you sign off the netlist.

You must check that you are meeting your setup and hold constraints.

16.3 Completion of sign-off

For successful completion of sign-off, you must generate and retain completed and verified Arm-related deliverables from the implementation process.

Deliverables include:

- GDS-II output.
- All required reports and logs.