

**HB0800**

**MIV\_RV32IMA\_L1\_AHB v2.1**

**Handbook**

July 2018



---

a  **MICROCHIP** company

# 1 Revision History

---

The revision history describes the changes that were implemented in the document. The changes are listed by revision, starting with the most current publication.

## 1.1 Release 2.0

Updated for MIV\_RV32IMA\_L1\_AHB v2.1. This release adds cache with ECC, debug reset, and external debug halt features.

## 1.2 Release 1.0

Revision 1.0 was the first production-level publication of this document. Created for MIV\_RV32IMA\_L1\_AHB v2.0.

## Contents

---

<b>1</b>	<b>Revision History.....</b>	<b>2</b>
1.1	Release 2.0.....	2
1.2	Release 1.0.....	2
<b>2</b>	<b>Introduction .....</b>	<b>7</b>
2.1	Overview.....	7
2.2	Features .....	7
2.3	Core Version.....	7
2.4	Supported Families .....	7
2.5	Device Utilization and Performance .....	8
<b>3</b>	<b>Functional Description .....</b>	<b>9</b>
3.1	MIV_RV32IMA_L1_AHB Architecture.....	9
3.2	MIV_RV32IMA_L1_AHB Processor Core.....	10
3.3	Pipelined Architecture .....	10
3.4	Memory System.....	11
3.5	Platform-Level Interrupt Controller .....	11
3.6	Debug support through JTAG.....	11
3.7	External AHB Interfaces .....	12
3.8	External HALT CPU .....	12
3.9	ECC.....	12
<b>4</b>	<b>Interface .....</b>	<b>13</b>
4.1	Configuration Parameters .....	13
4.1.1	MIV_RV32IMA_L1_AHB Configurable Options.....	13
4.1.2	Signal Descriptions .....	13
<b>5</b>	<b>Memory Map and Descriptions .....</b>	<b>16</b>
<b>6</b>	<b>Tool Flow .....</b>	<b>17</b>
6.1	License .....	17
6.1.1	RTL .....	17
6.2	SmartDesign.....	17
6.3	Configuring MIV_RV32IMA_L1_AHB in SmartDesign .....	17
6.4	Debugging.....	18
6.5	Simulation Flows.....	18
6.6	Synthesis in Libero .....	19
6.7	Place-and-Route in Libero.....	19
<b>7</b>	<b>System Integration .....</b>	<b>20</b>
7.1	Example System .....	20
7.2	Reset Synchronization.....	20

7.2.1	RST .....	20
7.2.2	TRST .....	21
8	Design Constraints .....	22
9	SoftConsole .....	24
10	Known Issues.....	25
10.1	Reset/Power Cycle the Target Hardware before each Debug Session .....	25

## Figures

---

Figure 1 MIV_RV32IMA_L1_AHB Block Diagram.....	10
Figure 2 Example Five Stage Pipelined Architecture .....	11
Figure 3 SmartDesign MIV_RV32IMA_L1_AHB Instance View .....	17
Figure 4 Configuring MIV_RV32IMA_L1_AHB in SmartDesign .....	18
Figure 5 Enabling Single Event Transient migration .....	18
Figure 6 RTG4 Example Simulation Subsystem.....	19
Figure 7 MIV_RV32IMA_L1_AHB Example System.....	20
Figure 8 RST Reset Synchronization.....	21

## Tables

---

Table 1 Device Utilization and Performance .....	8
Table 2 MIV_RV32IMA_L1_AHB Architecture .....	9
Table 3 Example Pipeline Timing .....	11
Table 4 MIV_RV32IMA_L1_AHB Configuration Options .....	13
Table 5 MIV_RV32IMA_L1_AHB I/O Signals .....	13
Table 6 Physical Memory Map (from Rocket-Chip) .....	16

---

## 2 Introduction

---

### 2.1 Overview

The MIV\_RV32IMA\_L1\_AHB is a softcore processor designed to implement the RISC-V instruction set for use in Microsemi FPGAs. The processor is based on Rocket-Chip, an open source high-performance single-issue, in-order execution pipeline 32-bit RISC-V core. The core includes an industry-standard JTAG interface to facilitate debug access, along with separate AHB bus interfaces for memory access and support for 31 dedicated interrupt ports. This core includes optional features such as ECC on the instruction and data cache and an external processor halt port for debug.

A quick start guide is available on how to create a MIV design from

<https://www.microsemi.com/product-directory/fpgas-socs-training/4339-fpga-training-tutorials>

### 2.2 Features

- Designed for FPGA soft-core implementation.
- Integrated 8Kbytes instructions cache and 8Kbytes data cache.
- A Platform-Level Interrupt Controller (PLIC) can support up to 31 programmable interrupts with a single priority level.
- Supports the RISC-V standard RV32IMA ISA.
- On-Chip debug unit with a JTAG interface.
- Two external AHB interfaces for IO and memory.
- External HALT signal can halt processor execution during a debug session.
- Support for Error-Correcting Code (ECC) caches on RTG4 and PolarFire.

### 2.3 Core Version

This Handbook applies to MIV\_RV32IMA\_L1\_AHB version 2.1.

**Note:** There are two accompanying manuals for this core:

- The RISC-V Instruction Set Manual, Volume 1, User Level ISA, Version 2.2
- The RISC-V Instruction Set Manual, Volume 2, Privileged Architecture, Version 1.10

### 2.4 Supported Families

- PolarFire®
- RTG4™
- IGLOO®2
- SmartFusion®2

## 2.5 Device Utilization and Performance

Utilization and performance data is listed in [Table 1](#) for the supported device families. The data listed in this table is indicative only. The overall device utilization and performance of the core is system dependent.

**Table 1 Device Utilization and Performance**

Device Family	Device	Sequential	Combinatorial	Max Frequency	Min Timing	Max Timing
IGLOO2	M2GL050	5080	10946	85.0MHz	Pass	Pass
PolarFire	MPF300TS	5096	11073	120.0Mhz	Pass	Pass
RTG4	RT4G150	5172	11134	66.0MHz	Pass	Pass
SmartFusion2	M2S090	5080	10946	85.0MHz	Pass	Pass

**Notes:**

1. Device Utilization and Performance numbers recorded with retiming enabled and with multi-pass set to 5.
2. All devices were built for standard speed grade apart from PolarFire, which was checked at -1. ECC enabled on PolarFire and RTG4.

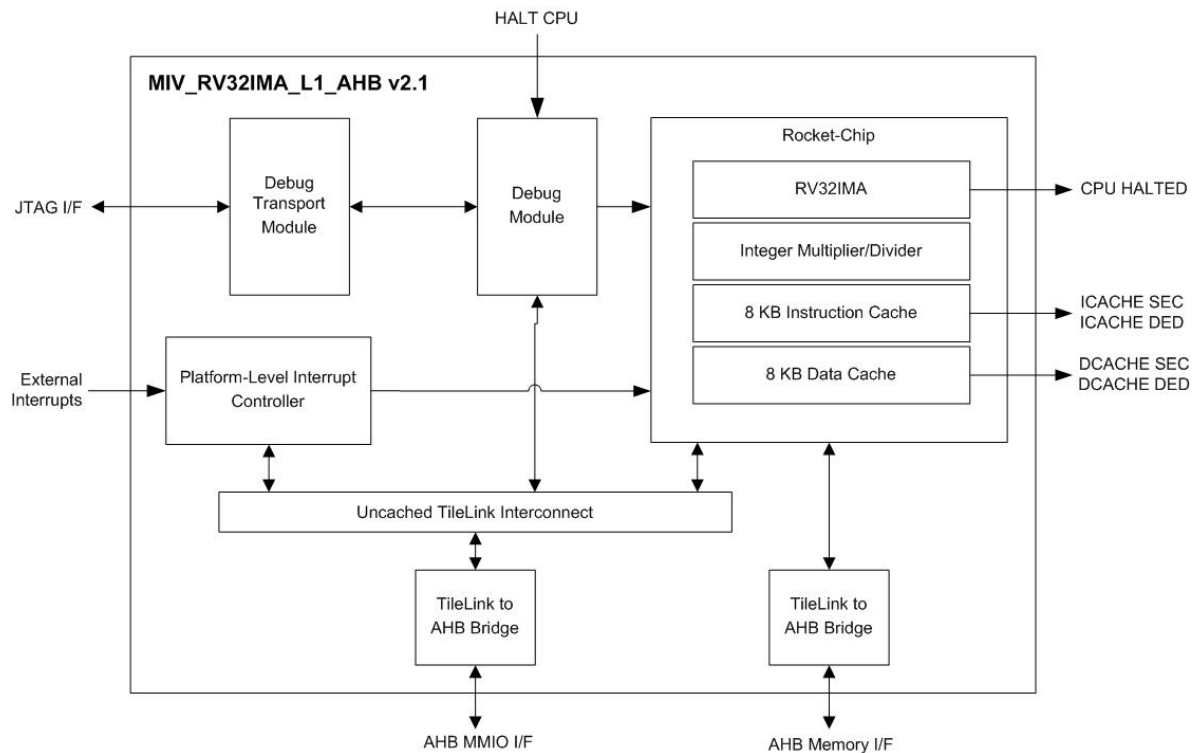


## 3 Functional Description

### 3.1 MIV\_RV32IMA\_L1\_AHB Architecture

**Table 2 MIV\_RV32IMA\_L1\_AHB Architecture**

Parameter	Value	Units	Notes
ISA Support	RV32IMA		
Cores	1		
Harts/Cores	1		
Branch prediction	None		Static Not Taken
Multiplier occupancy	16	cycles	2-bit/cycles iterative multiply
I-cache size	8	KiB	direct-mapped
I-cache associativity	1	way	
I-cache line-size	64	bytes	
D-cache size	8	KiB	direct-mapped
D-cache associativity	1	way	
D-cache line-size	64	bytes	
Reset Vector	configurable		
External interrupts	31		Fixed priorities
PLIC Interrupt priorities	1		
External memory bus	AHB		
External I/O bus	AHB		
JTAG debug transport address width	7	bits	
Hardware breakpoints	2		

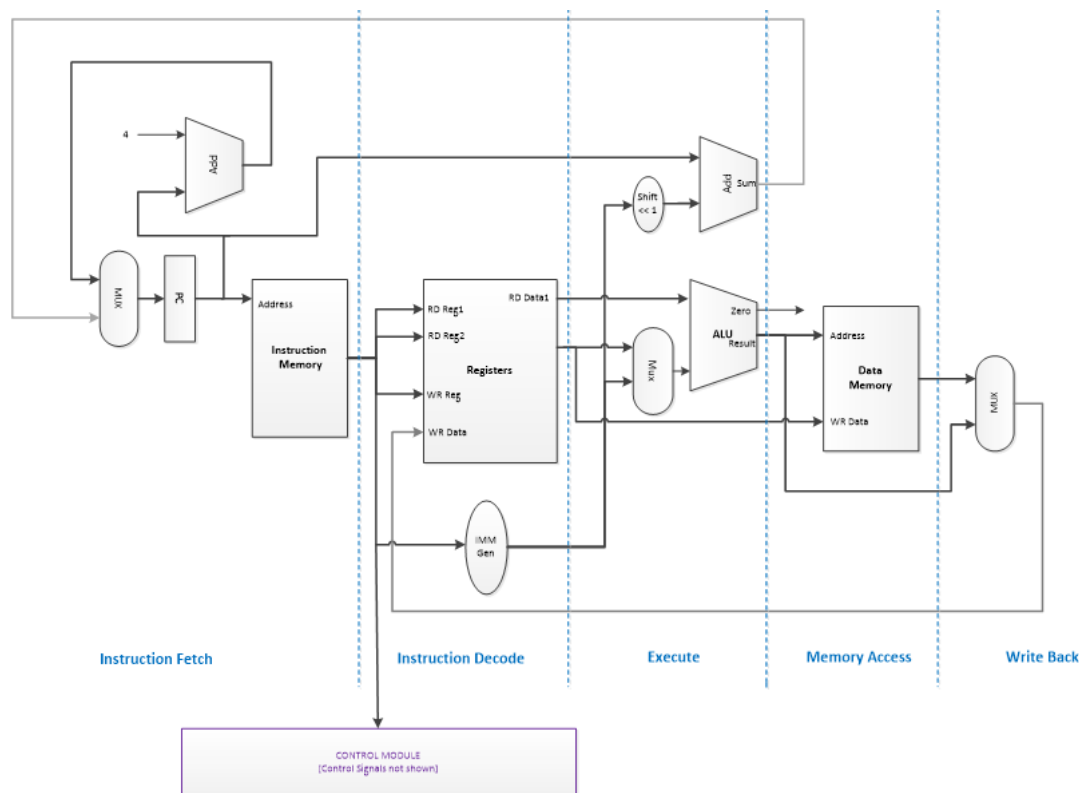
**Figure 1 MIV\_RV32IMA\_L1\_AHB Block Diagram**

### 3.2 MIV\_RV32IMA\_L1\_AHB Processor Core

MIV\_RV32IMA\_L1\_AHB is based on a Rocket-Chip implementation. The core provides a single hardware thread (or hart) supporting the RISC-V standard RV32IMA ISA and machine-mode privileged architecture.

### 3.3 Pipelined Architecture

MIV\_RV32IMA\_L1\_AHB provides a high-performance single-issue in-order 32-bit execution pipeline, with a peak sustainable execution rate of one instruction per clock cycle. The RISC-V ISA standard M extensions add hardware multiply and divide instructions. MIV\_RV32IMA\_L1\_AHB has a range of performance options including a fully pipelined multiply unit. An example of the pipeline and its timing is shown below.

**Figure 2 Example Five Stage Pipelined Architecture****Table 3 Example Pipeline Timing**

Clock Cycle	1	2	3	4	5	6	...	n
Fetch	Instruction 1	Instruction 2	Instruction 3	Instruction 4	Instruction 5	Instruction 6	.....	Instruction n
Decode		Decode Instruction 1	Decode Instruction 2	Decode Instruction 3	Decode Instruction 4	Decode Instruction 5	.....	Decode Instruction n-1
Execute			Execute Instruction 1	Execute Instruction 2	Execute Instruction 3	Execute Instruction 4	.....	Execute Instruction n-2
Mem access				RD\WR Memory 1	Memory Access 2	Memory Access 3	.....	Memory Access n-3
Writeback					Write Back 1	Write Back 2	.....	Write Back n-4

### 3.4 Memory System

MIV\_RV32IMA\_L1\_AHB memory system supports configurable split first-level instruction and data caches with support for hardware cache flushing, as well as uncached memory accesses. An Error Correcting Code (ECC) option for the instruction and data cache is available for PolarFire and RTG4 families.

### 3.5 Platform-Level Interrupt Controller

MIV\_RV32IMA\_L1\_AHB includes a RISC-V standard platform-level interrupt controller (PLIC) configured to support up to 31 inputs with a single priority level.

### 3.6 Debug support through JTAG

MIV\_RV32IMA\_L1\_AHB includes full external debugger support over an industry-standard JTAG port, supporting two hardware breakpoints.

### 3.7 External AHB Interfaces

MIV\_RV32IMA\_L1\_AHB includes two external AHB interfaces, bridged from the internal TileLink interfaces. The AHB memory interface is used by the cache controller to refill the instruction and data caches. The AHB I/O interface is used for uncached accesses to I/O peripherals. The cached range is from 0x8000\_0000 to 0x8FFF\_FFFF. The uncached range is 0x6000\_0000 to 0x7FFF\_FFFF.

The core can be booted from any aligned memory address on either the cached and uncached interface by setting the RESET\_VECTOR to the required boot memory address and modifying the linker scripts to match in the firmware project.

### 3.8 External HALT CPU

MIV\_RV32IMA\_L1\_AHB includes a parameter to enable an external halt CPU function to be used in conjunction with SoftConsole v5.2 that will, when the HALT\_CPU input is asserted, halt the CPU during a debug session and assert a CPU HALTED signal to confirm a halt has taken place. Please note that the CPU HALTED signal will assert during a debug session for all events that halt the CPU, for example, breakpoints and so on.

### 3.9 ECC

MIV\_RV32IMA\_L1\_AHB includes optional Error-Correcting Code (ECC) functionality with Single Error Correction and Double Error Detection (SECDED) event flags. Once enabled the ECC caches are initialized and these signals will assert to alert the user that an error has been corrected or detected in the instruction or data cache.

## 4 Interface

### 4.1 Configuration Parameters

#### 4.1.1 MIV\_RV32IMA\_L1\_AHB Configurable Options

There are four configurable options that apply to MIV\_RV32IMA\_L1\_AHB as shown in [Table 4](#). If a configuration other than the default is required, use the configuration dialog box in SmartDesign to select appropriate values for the configurable options.

**Table 4 MIV\_RV32IMA\_L1\_AHB Configuration Options**

Parameter	Valid Range	Default	Description
RESET_VECTOR_ADDR (Upper 16 bits [Hex])	0x6000 - 0x8FFF	0x6000	This is the address the processor will start executing from after a reset. This address is byte aligned.
RESET_VECTOR_ADDR (Lower 16 bits [Hex])	0x0000 – 0xFFFC	0x0	
EXT_HALT	Y/N	N	This will expose the “HALT_CPU” input pin and the “CPU_HALTED” output pin.
ECC_EN	Y/N	N	This will enable ECC and will expose the SECCED pins for both instruction and data caches.

#### 4.1.2 Signal Descriptions

Signal descriptions for MIV\_RV32IMA\_L1\_AHB are defined in [Table 5](#).

**Table 5 MIV\_RV32IMA\_L1\_AHB I/O Signals**

Port Name	Width	Direction	Description
<b>Global Signals</b>			
CLK	1	Input	System clock. All other I/Os are synchronous to this clock.
RESETN	1	Input	Synchronous reset signal. Active Low.
EXT_RESETN	1	Output	External Reset, which can be used to reset peripherals in a SoC design. It allows peripherals to be reset at commencement of a Debug session.
<b>JTAG Interface Signals</b>			
TDI	1	Input	Test Data In (TDI). This signal is used by the JTAG device for downloading and debugging programs. Sampled on the rising edge of TCK.
TCK	1	Input	Test Clock (TCK). This signal is used by the JTAG device for downloading and debugging programs.
TMS	1	Input	Test Mode Select (TMS). This signal is used by the JTAG device when downloading and debugging programs. It is sampled on the rising edge of TCK to determine the next state.
TRST	1	Input	Test Reset (TRST). This is an optional signal used to reset the TAP controllers state machine.

Port Name	Width	Direction	Description
TDO	1	Output	Test Data Out (TDO). This signal is the data, which is shifted out of the device during debugging. It is valid on FALLING/RISING edge of TCK.
DRV_TDO	1	Output	Drive Test Data Out (DRV_TDO). This signal is used to drive a tristate buffer.
External Interrupts Signals			
IRQ	31	Input	External interrupts from off-chip or peripheral sources. These are level-based interrupt signals.
Parameterized Signals			
HALT_CPU	1	Input	The “HALT_CPU” pin is active high and can be used to HALT the CPU during a debug session.
CPU_HALTED	1	Output	The “CPU_HALTED” pin will assert when the CPU has been successfully halted.
ICACHE_SEC	1	Output	The ICACHE_SEC pin will assert if a single error correction has occurred in the instruction cache.
ICACHE_DED	1	Output	The ICACHE_DED pins will assert if a double error detection has occurred in the instruction cache.
DCACHE_SEC	1	Output	The DCACHE_SEC pin will assert if a single error correction has occurred in the data cache.
DCACHE_DED	1	Output	The DCACHE_DED pins will assert if a double error detection has occurred in the data cache.
AHB Cached Memory Bus Master Interface			
AHB_MST_MEM_HLOCK	1	Output	AHB Master interface for cached memory accesses.
AHB_MST_MEM_HTRANS	2	Output	
AHB_MST_MEM_HSEL	1	Output	
AHB_MST_MEM_HWRITE	1	Output	
AHB_MST_MEM_HADDR	32	Output	
AHB_MST_MEM_HSIZE	3	Output	
AHB_MST_MEM_HBURST	3	Output	
AHB_MST_MEM_HPROT	4	Output	
AHB_MST_MEM_HWDATA	32	Output	
AHB_MST_MEM_HREADY	1	Input	
AHB_MST_MEM_HRESP	1	Input	
AHB_MST_MEM_HRDATA	32	Input	
AHB Non-cached Memory Bus Interface			
AHB_MST_MMIO_HLOCK	1	Output	AHB Master Interface for non-cached memory accesses.
AHB_MST_MMIO_HTRANS	2	Output	
AHB_MST_MMIO_HWRITE	1	Output	
AHB_MST_MMIO_HADDR	31	Output	
AHB_MST_MMIO_HSIZE	3	Output	
AHB_MST_MMIO_HBURST	3	Output	
AHB_MST_MMIO_HPROT	4	Output	

Port Name	Width	Direction	Description
AHB_MST_MMIO_HWDATA	32	Output	
AHB_MST_MMIO_HREADY	1	Input	
AHB_MST_MMIO_HRESP	1	Input	
AHB_MST_MMIO_HRDATA	32	Input	

---

## 5 Memory Map and Descriptions

---

**Table 6 Physical Memory Map (from Rocket-Chip)**

Base	Top	Description
0x0000_0000	0x0000_1000	Debug Controller
0x0000_3000	0x0000_4000	Error Device
0x4000_0000	0x4400_0000	Platform-Level Interrupt Control (PLIC)
0x4400_0000	0x4401_0000	Core Local Interrupt (CLINT)
0x6000_0000	0x7FFF_FFFF	AHB uncached interface
0x8000_0000	0x8FFF_FFFF	AHB cached Interface



## 6 Tool Flow

### 6.1 License

This core is being released under a modified Apache 2.0 license and is freely available through Libero.

#### 6.1.1 RTL

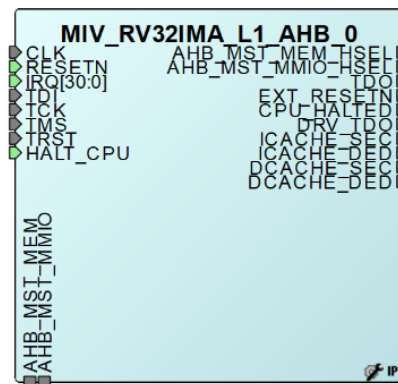
Complete Verilog source code is provided for the core. The core can be instantiated in Verilog or VHDL projects with SmartDesign. Simulation, Synthesis, and Layout performed within Libero SoC v11.8 or later.

### 6.2 SmartDesign

MIV\_RV32IMA\_L1\_AHB is preinstalled in SmartDesign IP Deployment design environment.

For more information on using SmartDesign to instantiate and generate cores, refer to the [Using DirectCore in Libero® SoC User Guide](#).

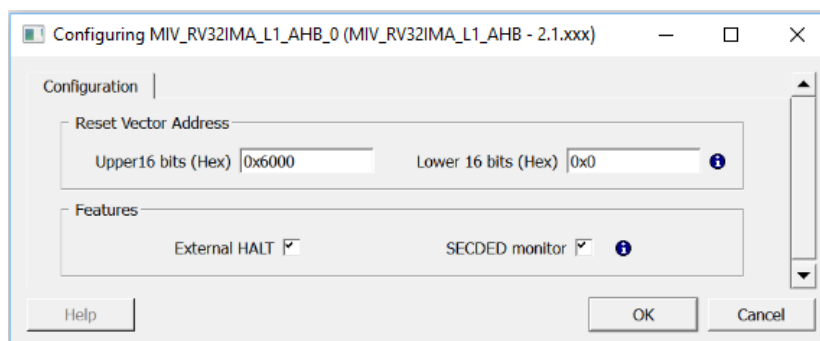
**Figure 3 SmartDesign MIV\_RV32IMA\_L1\_AHB Instance View**



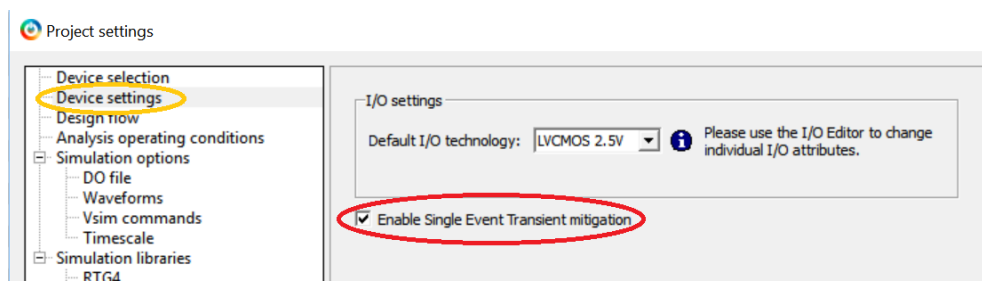
### 6.3 Configuring MIV\_RV32IMA\_L1\_AHB in SmartDesign

The core is configured using the configuration GUI within SmartDesign, as shown in [Figure 4](#).

**Note:** Leading zeros are suppressed, for example, 0x6000 0000 is displayed as 0x6000 0x0. The reset vector is byte aligned. Also, note that the SECEDED option is only selectable for RTG4 and PolarFire designs.

**Figure 4 Configuring MIV\_RV32IMA\_L1\_AHB in SmartDesign**

For RTG4 designs, the **Enable Single Event Transient Migration** feature in the **Device Settings** section of the Libero project settings can be enabled, as shown in [Figure 5](#).

**Figure 5 Enabling Single Event Transient migration**

## 6.4 Debugging

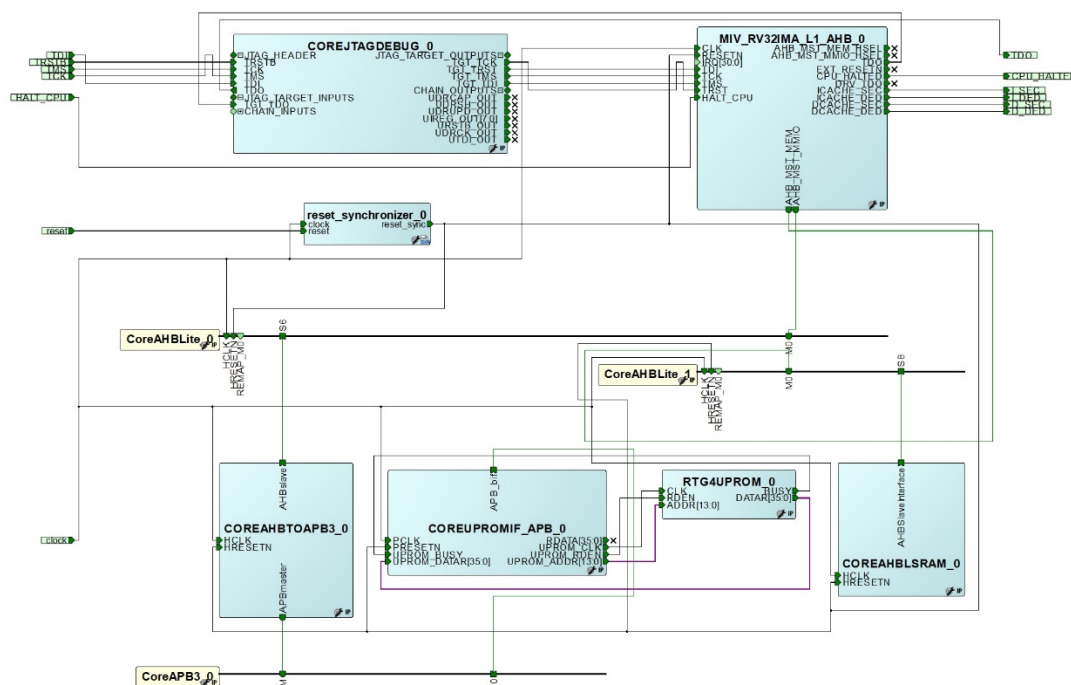
CoreJTAGDebug V2.0.100 or later, is used to enable debugging of MIV\_RV32IMA\_L1\_AHB. This is available in the Libero Catalog.

## 6.5 Simulation Flows

The user testbench for MIV\_RV32IMA\_L1\_AHB is not included in this release.

The MIV\_RV32IMA\_L1\_AHB RTL can be used to simulate the processor executing a program using a standard Libero generated HDL testbench. An example subsystem for RTG4 is as shown in [Figure 6](#).

**Figure 6 RTG4 Example Simulation Subsystem**



## 6.6 Synthesis in Libero

To run synthesis on the core, set the SmartDesign sheet as the design root and click **Synthesis** in Libero SoC.

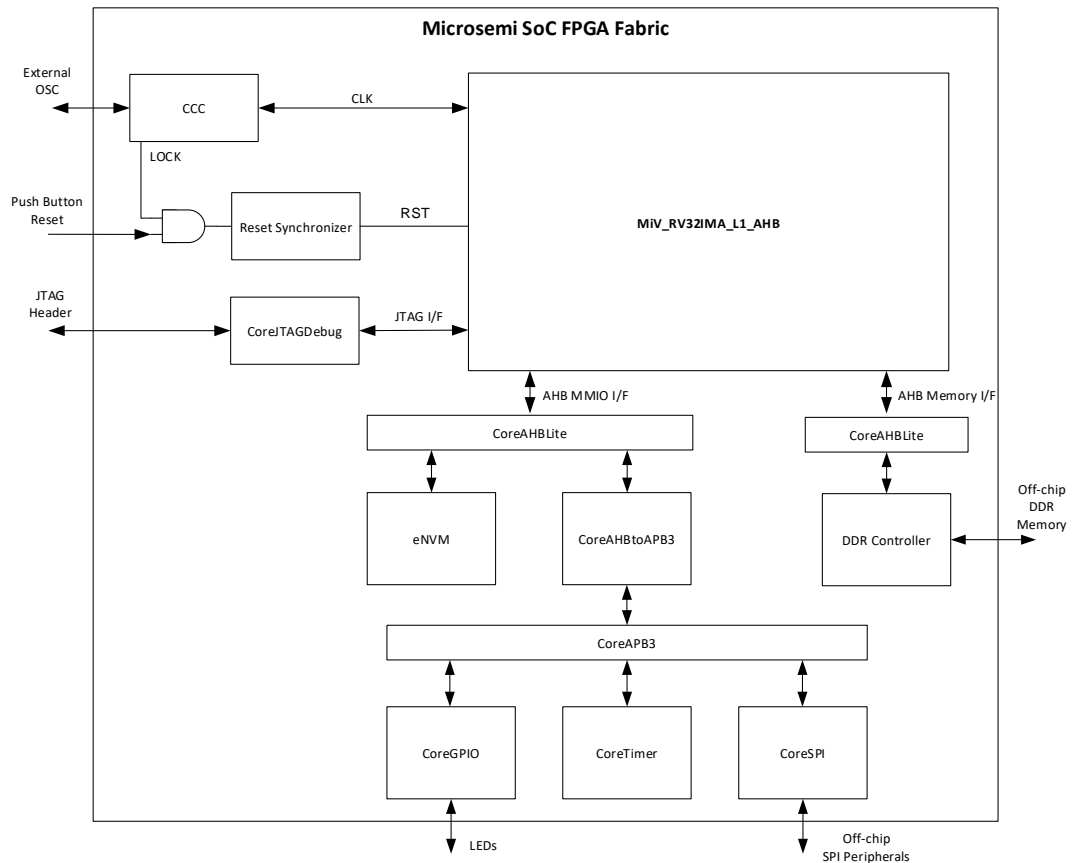
## 6.7 Place-and-Route in Libero

After the design is synthesized, run the compilation and the place and-route tools. Click **Layout** in the Libero SoC to invoke Designer. MIV\_RV32IMA\_L1\_AHB requires the place-and-route multi-seed settings set to 5.

## 7 System Integration

### 7.1 Example System

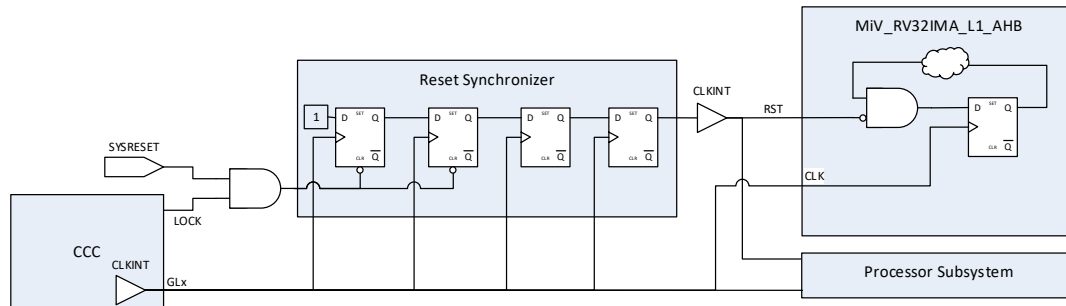
Figure 7 MIV\_RV32IMA\_L1\_AHB Example System



### 7.2 Reset Synchronization

#### 7.2.1 RST

All sequential elements clocked by **CLK** within **MIV\_RV32IMA\_L1\_AHB**, which require a reset employ a synchronous reset topology. Since, most designs source **CLK** from a CCC/PLL, it is common practice to AND the **LOCK** output of the CCC with the push button reset to generate the **RST** input for **MIV\_RV32IMA\_L1\_AHB**. However, this results in the reset being deasserted when the **CLK** comes up, hence the reset assertion is not clocked through the sequential reset elements and goes unnoticed most commonly leading to the processor locking-up. To guarantee that the **RST** assertion is seen by all sequential elements, a reset synchronizer is required on the **RST** input, as shown in [Figure 8](#).

**Figure 8 RST Reset Synchronization**

The Verilog code snippet below implements the reset synchronizer block as shown in [Figure 8](#). The function of this block is to make the reset assertion and deassertion synchronous to CLK whilst guaranteeing that the reset will be seen asserted for one or more CLK cycles within MIV\_RV32IMA\_L1\_AHB to ensure that it is registered by all sequential elements.

```

module reset_synchronizer (
    input clock,
    input reset,
    output reset_sync
);
    reg [1:0] sync_deassert_reg;
    reg [1:0] sync_asert_reg;

    always @ (posedge clock or negedge reset)
    begin
        if (!reset)
        begin
            sync_deassert_reg[1:0] <= 2'b00;
        end
        else
        begin
            sync_deassert_reg[1:0] <= {sync_deassert_reg[0], 1'b1};
        end
    end

    always @ (posedge clock)
    begin
        sync_asert_reg[1:0] <= {sync_asert_reg[0], sync_deassert_reg[1]};
    end
    assign reset_sync = sync_asert_reg[1];

endmodule

```

To include this synchronizer in your Libero design, select Create HDL from the Design Flow tab in your Libero project. In the popup window, name the HDL file accordingly and select Verilog as the HDL type whilst unchecking the option to Initialize file with standard template. Copy and paste the Verilog code snippet above into this file and save the changes. From the Design Hierarchy tab drag and drop the file into the SmartDesign sheet containing the MIV\_RV32IMA\_L1\_AHB instance and connect up the pins as shown above.

## 7.2.2 TRST

No reset synchronization is required on this reset input as all sequential elements in the debug logic within MIV\_RV32IMA\_L1\_AHB use an asynchronous reset topology.

## 8 Design Constraints

Designs containing MIV\_RV32IMA\_L1\_AHB require the application of the following constraints in the design flow to allow timing-driven placement and static timing analysis to be performed on MIV\_RV32IMA\_L1\_AHB. The procedure for adding the required constraints in the Enhanced Constraints flow in Libero v11.7 or later is as follows:

1. Double-click **Constraints > Manage Constraints** in the **Design Flow** window and click the **Timing** tab.

Assuming that the system clock used to clock MIV\_RV32IMA\_L1\_AHB is sourced from a PLL, select **Derive** to automatically create a constraints file containing the PLL constraints. Select **Yes** when prompted to allow the constraints to be automatically included for Synthesis, Place-and-Route, and Timing Verification stages.

If changes are made to the PLL configuration in the design, update the contents of this file by clicking **Derive**. Select **Yes** when prompted to allow the constraints to be overwritten.

2. In the **Timing** tab of the **Constraint Manager** window, select **New** to create a new SDC file, and name it. Design constraints other than the system clock source derived constraints can be entered in this blank SDC file. Keeping derived and manually added constraints in separate SDC files allows the **Derive** stage to be reperformed if changes are made to the PLL configuration, without deleting all manually added constraints in the process.
3. Calculate the TCK period and half period. TCK is typically 6 MHz when debugging with FlashPro, with a maximum frequency of 30 MHz supported by FlashPro5. After completion, enter the following constraints in the blank SDC file:

```
create_clock -name { TCK } \
  -period TCK_PERIOD \
  -waveform { 0 TCK_HALF_PERIOD } \
  [ get_ports { TCK } ]
```

For example, the following constraints need to be applied for a design that uses a TCK frequency of 6 MHz:

```
create_clock -name { TCK } \
  -period 166.67 \
  -waveform { 0 83.33 } \
  [ get_ports { TCK } ]
```

4. Next constraints must be applied to paths crossing the clock domain crossing between the TCK and system clock clock domains. MIV\_RV32IMA\_L1\_AHB implements two clock domain crossing FIFOs to handle the CDC and as such paths between the two clock domains may be declared as false paths to prevent min and max violations from being reported by SmartTime.

```
set_false_path -from [ get_clocks { TCK } ] \
  -to [ get_clocks { PLL_GEN_CLK } ]

set_false_path -from [ get_clocks { PLL_GEN_CLK } ] \
  -to [ get_clocks { TCK } ]
```

Where:

- PLL\_GEN\_CLK is the name applied to the create\_generated\_clock constraint derived in step 1 above.
5. Associate all constraints files with the Synthesis, Place-and-Route and Timing Verification stages in the **Constraint Manager > Timing** tab by selecting the related check boxes for the SDC files in which the constraints were entered in.

---

## 9 SoftConsole

---

SoftConsole Version 5.2 or later is required to use MIV\_RV32IMA\_L1\_AHB. It should be noted that the optional external halt feature, EXT\_HALT works in conjunction with SoftConsole Version 5.2 and may not be operational in later SoftConsole releases. Each SoftConsole project requires the Hardware Abstraction Layer (HAL) version 2.1 or greater. The SoftConsole Release Notes details how to set up a project for the MIV\_RV32IMA\_L1\_AHB core.



---

## 10 Known Issues

---

### 10.1 Reset/Power Cycle the Target Hardware before each Debug Session

At the moment, the debugger cannot effect a suitable Mi-V RISC-V CPU/SoC reset at the start of each debug session so one debug session may be impacted by what went before – for example, a previous debug session leaves the CPU in an ISR and a subsequent debug session does not behave as expected because of this. To mitigate this problem, it is recommended that the target hardware/board is power cycled or otherwise reset before each new debug session.



**Microsemi Headquarters**  
 One Enterprise, Aliso Viejo, CA 92656 USA  
 Within the USA: +1 (800) 713-4113  
 Outside the USA: +1 (949) 380-6100  
 Sales: +1 (949) 380-6136  
 Fax: +1 (949) 215-4996  
 email: [sales.support@microsemi.com](mailto:sales.support@microsemi.com)  
[www.microsemi.com](http://www.microsemi.com)

©2018 Microsemi, a wholly owned subsidiary of Microchip Technology Inc. All rights reserved. Microsemi and the Microsemi logo are registered trademarks of Microsemi Corporation. All other trademarks and service marks are the property of their respective owners.

Microsemi, a wholly owned subsidiary of Microchip Technology Inc. (Nasdaq: MCHP), offers a comprehensive portfolio of semiconductor and system solutions for aerospace & defense, communications, data center and industrial markets. Products include high-performance and radiation-hardened analog mixed-signal integrated circuits, FPGAs, SoCs and ASICs; power management products; timing and synchronization devices and precise time solutions, setting the world's standard for time; voice processing devices; RF solutions; discrete components; enterprise storage and communication solutions, security technologies and scalable anti-tamper products; Ethernet solutions; Power-over-Ethernet ICs and midspans; as well as custom design capabilities and services. Learn more at [www.microsemi.com](http://www.microsemi.com).

Microsemi makes no warranty, representation, or guarantee regarding the information contained herein or the suitability of its products and services for any particular purpose, nor does Microsemi assume any liability whatsoever arising out of the application or use of any product or circuit. The products sold hereunder and any other products sold by Microsemi have been subject to limited testing and should not be used in conjunction with mission-critical equipment or applications. Any performance specifications are believed to be reliable but are not verified, and Buyer must conduct and complete all performance and other testing of the products, alone and together with, or installed in, any end-products. Buyer shall not rely on any data and performance specifications or parameters provided by Microsemi. It is the Buyer's responsibility to independently determine suitability of any products and to test and verify the same. The information provided by Microsemi hereunder is provided "as is, where is" and with all faults, and the entire risk associated with such information is entirely with the Buyer. Microsemi does not grant, explicitly or implicitly, to any party any patent rights, licenses, or any other IP rights, whether with regard to such information itself or anything described by such information. Information provided in this document is proprietary to Microsemi, and Microsemi reserves the right to make any changes to the information in this document or to any products and services at any time without notice.