HB0823

Handbook

MIV_RV32IMA_L1_AXI V2.0

03 2018

Microsemi

Power Matters.™

About Microsemi

Microsemi Corporation (Nasdaq: MSCC) 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. Microsemi is headquartered in Aliso Viejo, Calif., and has approximately 4,800 employees globally. Learn more at www.microsemi.com.

Microsemi Corporate 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
E-mail: sales.support@microsemi.com
www.microsemi.com

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 1.0

Revision 1.0 is the first publication of this document. Created for MIV_RV32IMA_L1_AXI v2.0.

Contents

List of Figures

List of Tables

2 Introduction

2.1 Overview

The MIV_RV32IMA_L1_AXI is a softcore processor designed to implement the RISC-V instruction set for use in Microsemi FPGAs. The processor is based on the Rocket-Chip RISC-V core. The core includes an industry-standard JTAG interface to facilitate debug access, along with separate AXI4 bus interfaces for memory access and support for 31 dedicated interrupt ports. 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 low power ASIC microcontroller and FPGA soft-core implementations.
- 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 RV-32IMA ISA.
- On-Chip debug unit with a JTAG interface.
- Optional AXI4 or pseudo AXI3 interfaces dedicated to peripheral IO and memory/data cache operations respectively

2.3 Core Version

This Handbook applies to MIV_RV32IMA_L1_AXI version 2.0.

Note: There are two accompanying manuals for this core:

- The RISC-V Instruction Set Manual, Volume 1, User Level ISA, Version 2.1
- 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	Sequential	Combinational	Max-Frequency	Min Timing	Max Timing
IGLOO2	6311	13160	86.0MHz	Pass	Pass
PolarFire	6302	12744	130.0Mhz	Pass	Pass
RTG4	6328	12883	63.0MHz	Pass	Pass
SmartFusion2	6314	13059	90.0MHz	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.

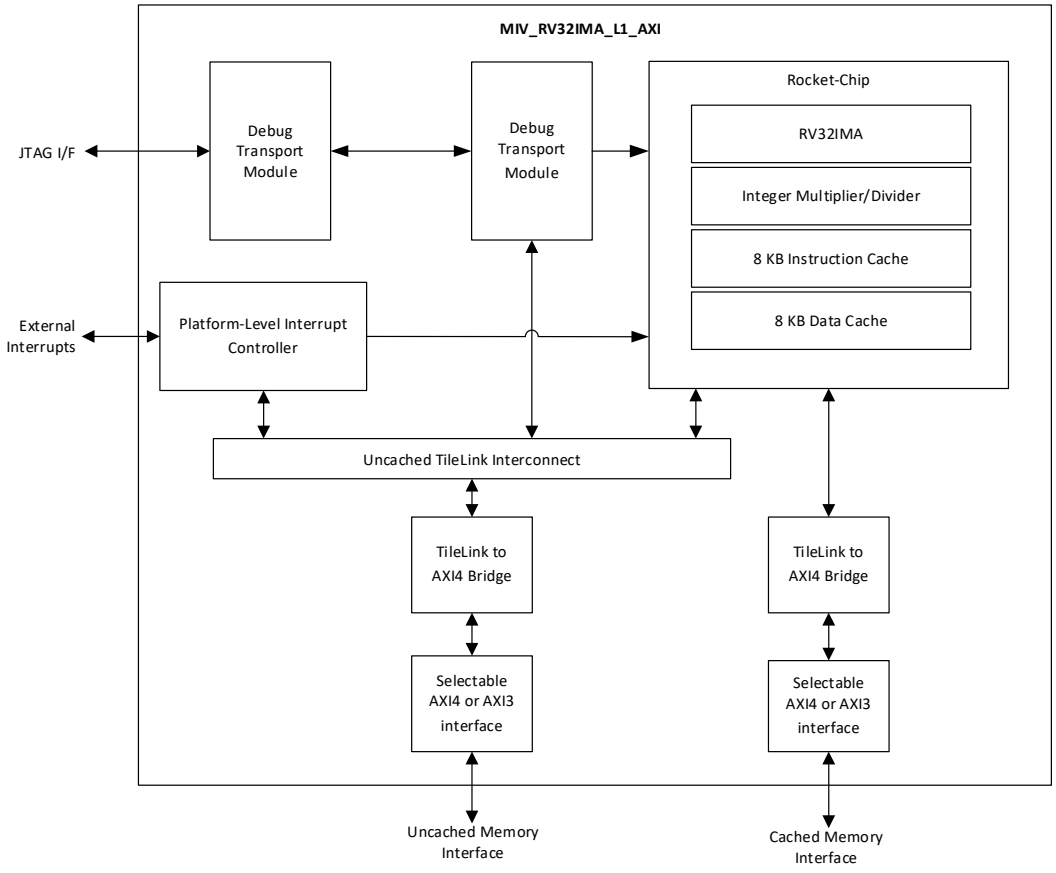
3 Functional Description

3.1 MIV_RV32IMA_L1_AXI Architecture

Table 2 MIV_RV32IMA_L1_AXI Architecture

Parameter	Value	Units	Notes
ISA Support	RV-32IMA		
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	
I-cache associativity	1	way	direct-mapped
I-cache line-size	64	bytes	
D-cache size	8	KiB	
D-cache associativity	1	way	direct-mapped
D-cache line-size	64	bytes	
Reset Vector	configurable		
External interrupts	31		
PLIC Interrupt priorities	1		Fixed priorities
External memory bus	AXI4/AXI3		Optional AXI4 or AXI3
External I/O bus	AXI4/AXI3		Optional AXI4 or AXI3
JTAG debug transport address width	7	bits	
Hardware breakpoints	2		

Figure 1 MIV_RV32IMA_L1_AXI Block Diagram



3.2 MIV_RV32IMA_L1_AXI Processor Core

MIV_RV32IMA_L1_AXI is based on the E31 Coreplex Core by SiFive. 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_AXI 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_AXI 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

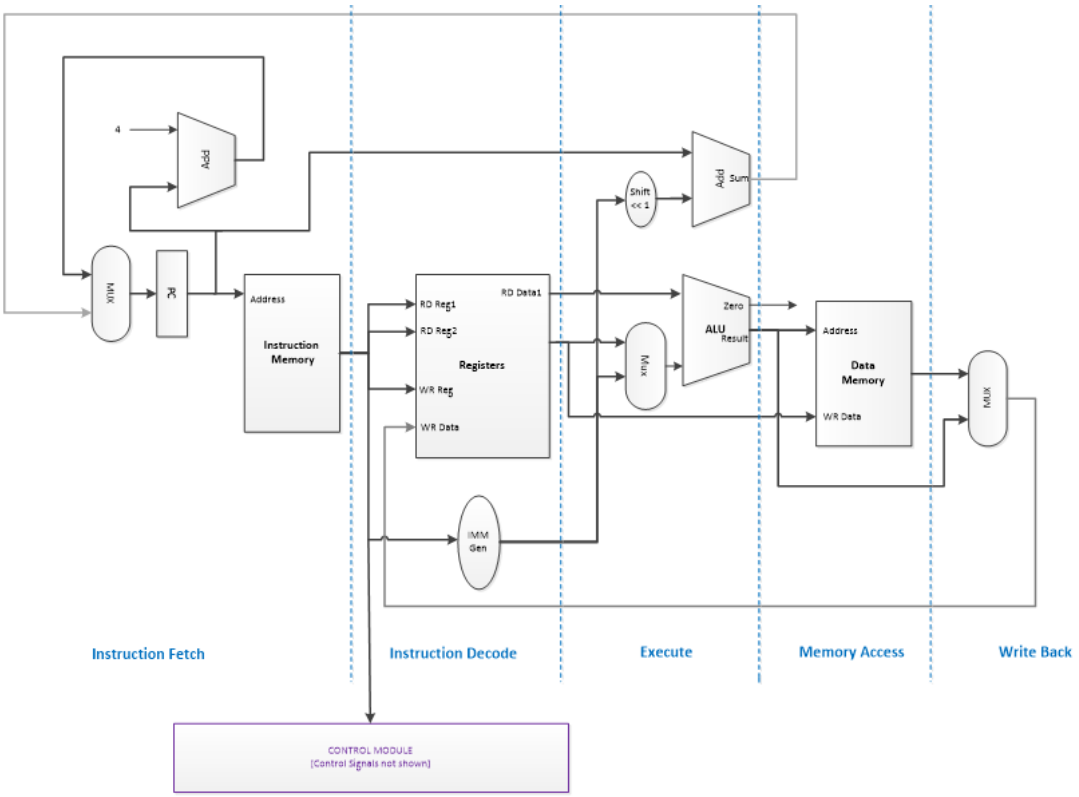


Table 3 Example Pipeline Timing

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_AXI memory system supports configurable split first-level instruction and data caches with full support for hardware cache flushing, as well as uncached memory accesses. External connections are provided for both cached and uncached TileLink fabrics.

3.5 Platform-Level Interrupt Controller

MIV_RV32IMA_L1_AXI includes a RISC-V standard platform-level interrupt controller (PLIC) configured to support up 31 inputs with a single priority level.

3.6 Debug Support via JTAG

MIV_RV32IMA_L1_AXI includes full external debugger support over an industry-standard JTAG port, supporting two hardware breakpoints.

3.7 External AXI4 Interfaces

MIV_RV32IMA_L1_AXI includes two external AXI4 interfaces, bridged from the internal TileLink interfaces. The AXI4 memory interface (AXI4_MST_MEM) is used by the cache controller to refill the instruction and data caches. The AXI I/O interface(AXI4_MST_MMIO) 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. Addresses on both the cached and uncached interfaces can be booted from by setting the RESET_VECTOR and modifying the linker scripts for the firmware project.

A pseudo AXI3 interface is available. This interface converts the AXI4 interface on the cached and uncached interface to AXI3.

4 Interface

4.1 Configuration Parameters

4.1.1 MIV_RV32IMA_L1_AXI Configurable Options

There are several configurable options that apply to MIV_RV32IMA_L1_AXI 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.

The MIV_RV32IMA_L1_AXI core has two cache interfaces, cached (AXI4_MST_MEM) and uncached (AXI4_MST_MMIO). The cached range is from 0x8000_0000 to 0x8FFF_FFFF. The uncached range is 0x6000_0000 to 0x7FFF_FFFF.

Table 4 MIV_RV32IMA_L1_AXI Configuration Options

Parameter	Valid Range	Default	Description
RESET_VECTOR_ADDR_1 (high halfword)	0x6000 - 0x8FFF	0x6000	This is the address the processor will start executing from after a reset. This address is word aligned. Opted for underscore to illustrate parameter concatenation. Please note upper Reset Vector Address boundary is at 0x8FFF_FFFF.
RESET_VECTOR_ADDR_0 (low halfword)	0x0000 – 0xFFFC	0x0	
MASTER_TYPE	AXI4 OR AXI3	AXI3	Selects either AXI4 or pseudo AXI3 interfaces for the cached and uncached interfaces
MEM_WID	0 to 31	6	System Write ID tag value (specific to AXI3)
MMIO_WID	0 to 31	9	System Write ID tag value (specific to AXI3)

4.1.2 Signal Descriptions

Signal descriptions for MIV_RV32IMA_L1_AXI are defined in Table 5.

Table 5 MIV_RV32IMA_L1_AXI I/O Signals

Port Name	Width	Direction	Description
Global Signals			
CLK	1	In	System clock. All other I/Os are synchronous to this clock.
RESETN	1	In	Synchronous reset signal. Active Low
EXT_RESETN	1	Out	This is an optional active low reset output from the debug module which can be used to automatically reset SoC components or external devices during a debug session.
JTAG Interface Signals			
TDI	1	In	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	In	Test Clock (TCK). This signal is used by the JTAG device for downloading and debugging programs.
TMS	1	In	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	In	Test Reset (TRST). This is an optional signal used to reset the TAP controllers state machine.
TDO	1	Out	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	Out	Drive Test Data Out (DRV_TDO). This signal is used to drive a tristate buffer
External Interrupts Signals			
IRQ	31	In	External interrupts from off-chip or peripheral sources. These are level-based interrupt signals.

AXI4 Cached Memory Bus Master Interface			
Port Name	**Width**	**Direction**	**Description**
MEM_AXI_0_AW_READY	1	in	AXI4 Master Interface for cached memory accesses. Address range: 0x8000_0000 to 0x8FFF_FFFF.
MEM_AXI_0_AW_VALID	1	out	
MEM_AXI_0_AW_BITS_ID	4	out	
MEM_AXI_0_AW_BITS_ADDR	32	out	
MEM_AXI_0_AW_BITS_LEN	8	out	
MEM_AXI_0_AW_BITS_SIZE	3	out	
MEM_AXI_0_AW_BITS_BURST	2	out	
MEM_AXI_0_AW_BITS_LOCK	1	out	
MEM_AXI_0_AW_BITS_CACHE	4	out	
MEM_AXI_0_AW_BITS_PROT	3	out	
MEM_AXI_0_AW_BITS_QOS	4	out	
MEM_AXI_0_W_READY	1	in	
MEM_AXI_0_W_VALID	1	out	
MEM_AXI_0_W_BITS_DATA	64	out	
MEM_AXI_0_W_BITS_STRB	7	out	
MEM_AXI_0_W_BITS_LAST	1	out	
MEM_AXI_0_B_READY	1	out	
MEM_AXI_0_B_VALID	1	in	
MEM_AXI_0_B_BITS_ID	4	in	
MEM_AXI_0_B_BITS_RESP	2	in	
MEM_AXI_0_AR_READY	1	in	
MEM_AXI_0_AR_VALID	1	out	
MEM_AXI_0_AR_BITS_ID	3	out	
MEM_AXI_0_AR_BITS_ADDR	32	out	
MEM_AXI_0_AR_BITS_LEN	8	out	
MEM_AXI_0_AR_BITS_SIZE	3	out	
MEM_AXI_0_AR_BITS_BURST	2	out	
MEM_AXI_0_AR_BITS_LOCK	1	out	
MEM_AXI_0_AR_BITS_CACHE	4	out	
MEM_AXI_0_AR_BITS_PROT	3	out	
MEM_AXI_0_AR_BITS_QOS	4	out	
MEM_AXI_0_R_READY	1	out	
MEM_AXI_0_R_VALID	1	out	
MEM_AXI_0_R_BITS_ID	4	out	
MEM_AXI_0_R_BITS_DATA	64	out	

MEM_AXI_0_R_BITS_RESP	2	out	
MEM_AXI_0_R_BITS_LAST	1	out	
MEM_AXI_0_W_BITS_ID	5	out	This signal is included for compatibility with AXI3. Value defined in GUI
AXI4 Non-Cached Memory Bus Master Interface			
Port Name	**Width**	**Direction**	**Description**
MMIO_AXI_0_AW_READY	1	in	AXI4 Master Interface for non-cached memory accesses. Address range from 0x6000_0000 to 0x7FFF_FFFF.
MMIO_AXI_0_AW_VALID	1	out	
MMIO_AXI_0_AW_BITS_ID	4	out	
MMIO_AXI_0_AW_BITS_ADDR	31	out	
MMIO_AXI_0_AW_BITS_LEN	8	out	
MMIO_AXI_0_AW_BITS_SIZE	3	out	
MMIO_AXI_0_AW_BITS_BURST	2	out	
MMIO_AXI_0_AW_BITS_LOCK	1	out	
MMIO_AXI_0_AW_BITS_CACHE	4	out	
MMIO_AXI_0_AW_BITS_PROT	3	out	
MMIO_AXI_0_AW_BITS_QOS	4	out	
MMIO_AXI_0_W_READY	1	in	
MMIO_AXI_0_W_VALID	1	out	
MMIO_AXI_0_W_BITS_DATA	64	out	
MMIO_AXI_0_W_BITS_STRB	7	out	
MMIO_AXI_0_W_BITS_LAST	1	out	
MMIO_AXI_0_B_READY	1	out	
MMIO_AXI_0_B_VALID	1	in	
MMIO_AXI_0_B_BITS_ID	4	in	
MMIO_AXI_0_B_BITS_RESP	2	in	
MMIO_AXI_0_AR_READY	1	in	
MMIO_AXI_0_AR_VALID	1	out	
MMIO_AXI_0_AR_BITS_ID	4	out	
MMIO_AXI_0_AR_BITS_ADDR	31	out	
MMIO_AXI_0_AR_BITS_LEN	8	out	
MMIO_AXI_0_AR_BITS_SIZE	3	out	
MMIO_AXI_0_AR_BITS_BURST	2	out	
MMIO_AXI_0_AR_BITS_LOCK	1	out	
MMIO_AXI_0_AR_BITS_CACHE	4	out	
MMIO_AXI_0_AR_BITS_PROT	3	out	
MMIO_AXI_0_AR_BITS_QOS	4	out	

MMIO_AXI_0_R_READY	1	out	
MMIO_AXI_0_R_VALID	1	out	
MMIO_AXI_0_R_BITS_ID	4	out	
MMIO_AXI_0_R_BITS_DATA	64	out	
MMIO_AXI_0_R_BITS_RESP	2	out	
MMIO_AXI_0_R_BITS_LAST	1	out	
MMIO_AXI_0_W_BITS_ID	5	out	This signal is included for compatibility with AXI3. Value defined in GUI

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	AXI uncached interface
0x8000_0000	0x8FFF_FFFF	AXI cached Interface

Note: All other address ranges are reserved for future use.

6 Tool Flow

6.1 License

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

6.1.1 RTL

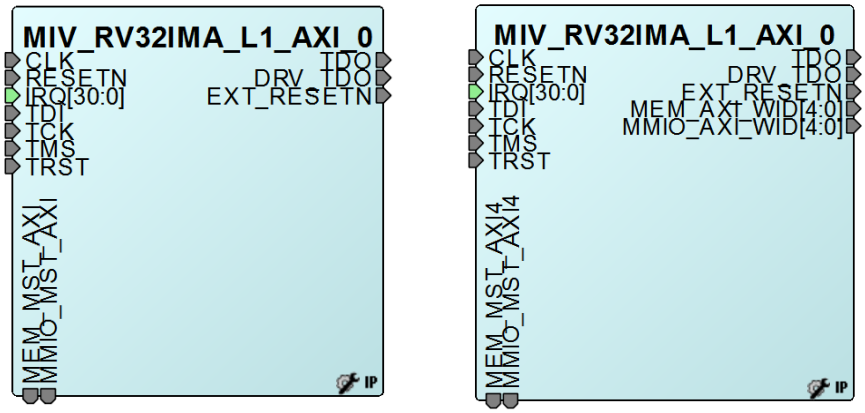
Complete Verilog source code is provided for the core. A VHDL wrapper is provided for use in VHDL projects. Allowing the core to be instantiated with SmartDesign. Simulation, Synthesis, and Layout can be performed within Libero SoC.

6.2 SmartDesign

MIV_RV32IMA_L1_AXI 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_AXI Instance Views



AXI3 Configuration

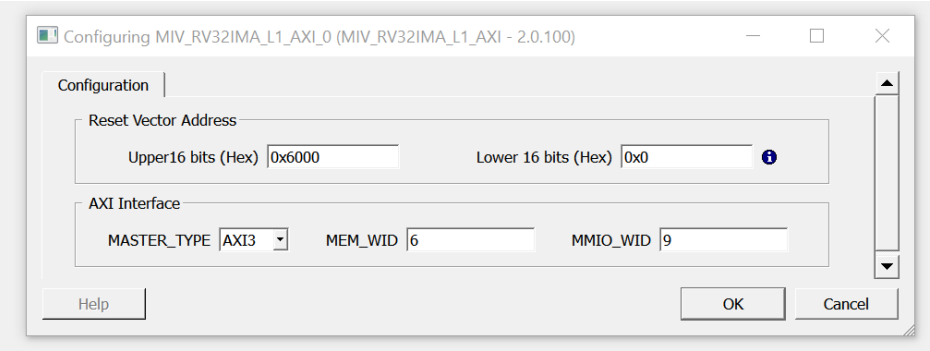
AXI4 Configuration

6.3 Configuring MIV_RV32IMA_L1_AXI 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 quad byte aligned.

Figure 4 Configuring MIV_RV32IMA_L1_AXI in SmartDesign



In Figure 4, the Reset Vector Address is 0x6000_0000, which is the default boot address. Code that runs from NVM must be built using the correct linker script. For example, in the RISC-V HAL there are two example linker scripts, Microsemi-riscv-ram.ld and Microsemi-riscv-ilgoo2.ld. The first is set up for Random Access Memories such as DDR or LSRAM. The second is set up for NVM. To boot from NVM on power up, the reset vector, linker script and hardware design must match. Otherwise, the design will not boot as expected.

An AXI3 or AXI4 Master Interface can be selected. MEM_WID and MMIO_WID values specific to an AXI3 based system can be modified if necessary. The default values shown are the values required for current Microsemi reference Libero designs.

6.4 Debugging

CoreJTAGDebug V2.0.100 or later, is used to enable debugging of MIV_RV32IMA_L1_AXI. This is available in the Libero Catalog.

6.5 Simulation Flows

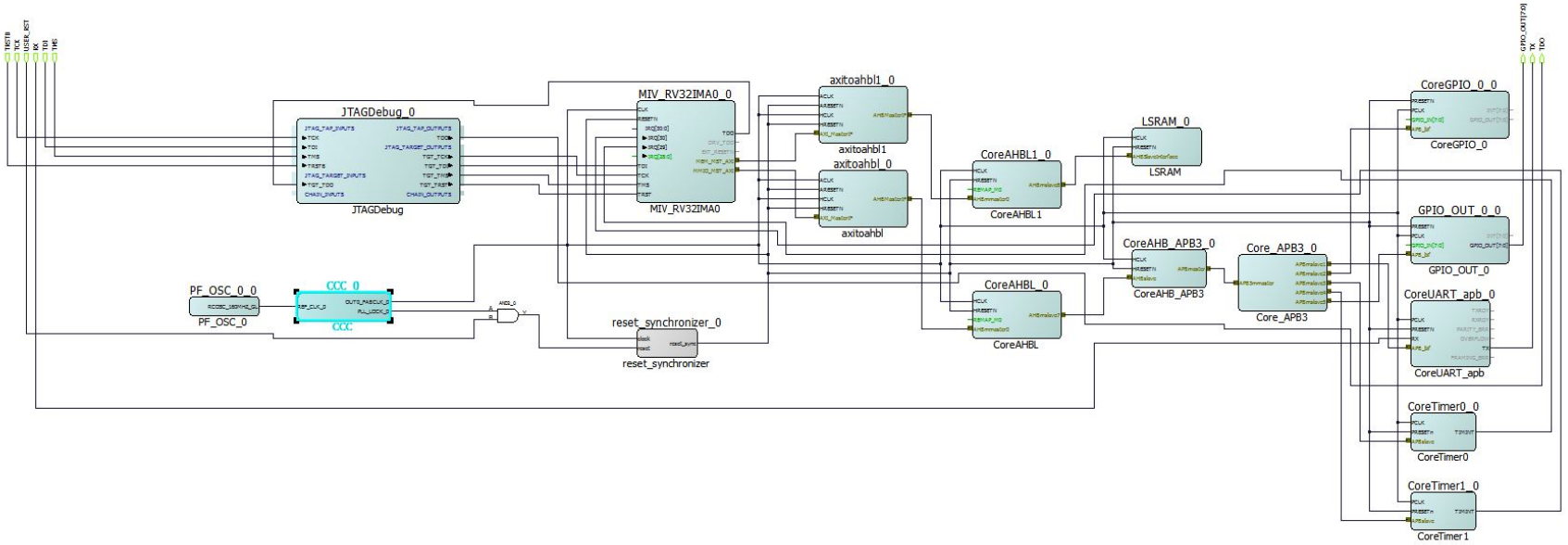
The user testbench for MIV_RV32IMA_L1_AXI is not included in this release.

The MIV_RV32IMA_L1_AXI RTL can be used to simulate the processor executing a program using a standard Libero generated HDL testbench. An example subsystem is shown in

Figure 5. A hex file found in the Debug or Release folders generated by SoftConsole V5.2 and above, is needed for this method. When the hex file is generated remove the first line before importing it into NVM. The RESET_VECTOR of the MIV core is set to 0x8000_0000 so it will boot from the LSRAM_0. Using this design the MIV core can be simulated.

The example system below is explained in the PolarFire tutorial https://www.microsemi.com/product-directory/fpgas-socs-training/4339-fpga-training-tutorials.

Figure 5 Example 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.

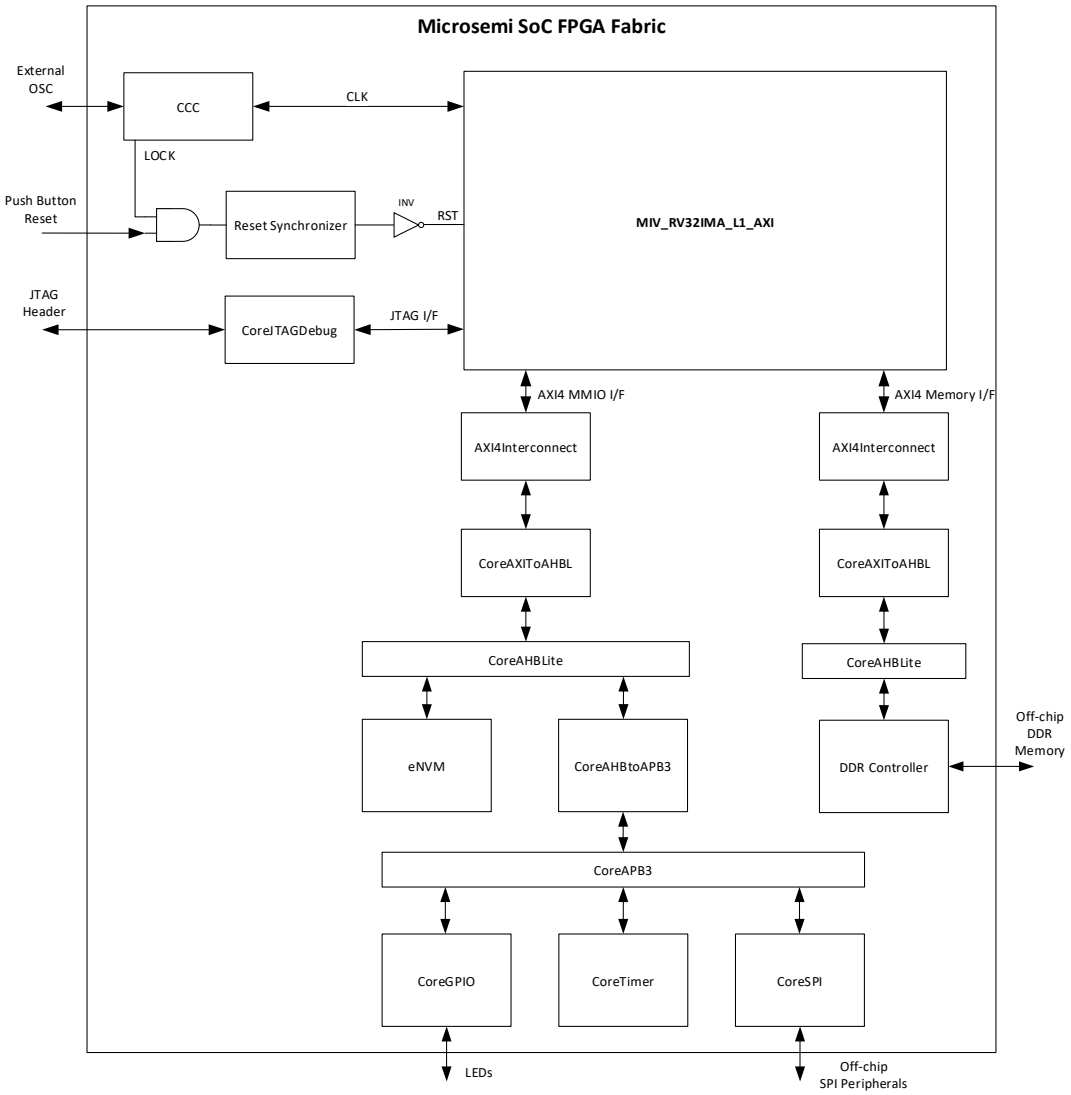
6.7 Place-and-Route in Libero

After the design is synthesized, run the compilation and the place and-route tools. Click the **Layout** icon in the Libero SoC to invoke Designer.

7 System Integration

7.1 Example System

Figure 6 MIV_RV32IMA_L1_AXI Example AXI4 System

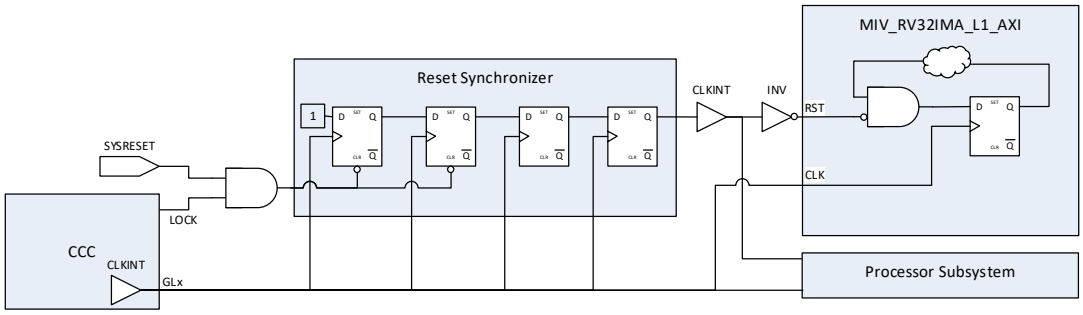


7.2 Reset Synchronization

7.2.1 RST

All sequential elements clocked by *CLK* within MIV_RV32IMA_L1_AXI, 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_AXI. 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 7.

Figure 7 RST Reset Synchronization



The Verilog code snippet below implements the reset synchronizer block shown in Figure 7. 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_AXI to ensure that it is registered by all sequential elements.

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

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


always @ (posedge clock)
    begin
        sync_asert_reg[1:0] <= {sync_asert_reg[0], sync_deasert_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_AXI instance and connect 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_AXI use an asynchronous reset topology.

8 Design Constraints

Designs containing MIV_RV32IMA_L1_AXI 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_AXI. 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_AXI 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_AXI 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 is required to use MIV_RV32IMA_L1_AXI. 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_AXI 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 – e.g. 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.