MicroBlaze Micro Controller System v2.3

LogiCORE IP Product Guide

Vivado Design Suite

PG116 June 24, 2015

XILINX
ALL PROGRAMMABLE™

Table of Contents

IP Facts

Chapter 1: Overview

Feature Summary 5
Licensing and Ordering Information 7

Chapter 2: Product Specification

Standards 8
Performance 8
Resource Utilization 8
Port Descriptions 9
Register Space 10

Chapter 3: Designing with the Core

General Design Guidelines 11
Clocking 11
Resets 12
Protocol Description 12

Chapter 4: Design Flow Steps

Customizing and Generating the Core 13
Constraining the Core 31
Simulation 31
Synthesis and Implementation 32

Appendix A: Migrating and Upgrading

Migrating to the Vivado Design Suite 33
Upgrading in the Vivado Design Suite 33

Appendix B: Debugging

Finding Help on Xilinx.com 34
Debug Tools 35
Simulation Debug 36

Hardware Debug 36

Appendix C: Application Software Development

Xilinx Software Development Kit 37
Device Drivers 37

Appendix D: Additional Resources and Legal Notices

Xilinx Resources 40
References 40
Revision History 41
Please Read: Important Legal Notices 41

Introduction

The LogiCORE™ IP MicroBlaze™ Micro Controller System (MCS) core is a complete processor system intended for controller applications. It is highly integrated and includes the MicroBlaze processor, local memory for program and data storage as well as a tightly coupled I/O module implementing a standard set of peripherals.

The MicroBlaze processor included in the MicroBlaze MCS core has a fixed configuration, optimized for minimal area. The full-featured MicroBlaze processor is available in the Vivado® Design Suite.

Features

- MicroBlaze processor
- Local Memory
- MicroBlaze Debug Module (MDM)
  - Debug UART
- Tightly Coupled I/O Module including
  - I/O Bus
  - Interrupt Controller using fast interrupt mode
  - UART
  - Fixed Interval Timers
  - Programmable Interval Timers
  - General Purpose Inputs
  - General Purpose Outputs

LogiCORE IP Facts Table	
Core Specifics	
Supported Device Family(1)	UltraScale™ Architecture, Zynq®-7000 All Programmable SoC, 7 Series
Supported User Interfaces	Local Memory Bus (LMB), Dynamic Reconfiguration Port (DRP)
Resources	Performance and Resource Utilization web page
Provided with Core	
Design Files	RTL
Example Design	Not Provided
Test Bench	Not Provided
Constraints File	Not Provided
Simulation Model	Verilog and/or VHDL Structural
Supported S/W Driver(2)	Standalone
Tested Design Flows(3)	
Design Entry	Vivado Design Suite
Simulation	For supported simulators, see the Xilinx Design Tools: Release Notes Guide.
Synthesis	Vivado Synthesis
Support	
Provided by Xilinx at the Xilinx Support web page	

Notes:

1. For a complete listing of supported devices, see the Vivado IP catalog.
2. Standalone driver details can be found in the SDK directory (<*install_directory*>/doc/usenglish/xilinx_drivers.htm). Linux OS and driver support information is available from the Xilinx Wiki page.
3. For the supported versions of the tools, see the Xilinx Design Tools: Release Notes Guide.

Overview

The MicroBlaze™ MCS core is a highly integrated processor system intended for controller applications. Data and program is stored in a local memory, debug is facilitated by the MicroBlaze Debug Module (MDM). A standard set of peripherals is also included, providing basic functionality like interrupt controller, UART, timers and general purpose input and outputs.

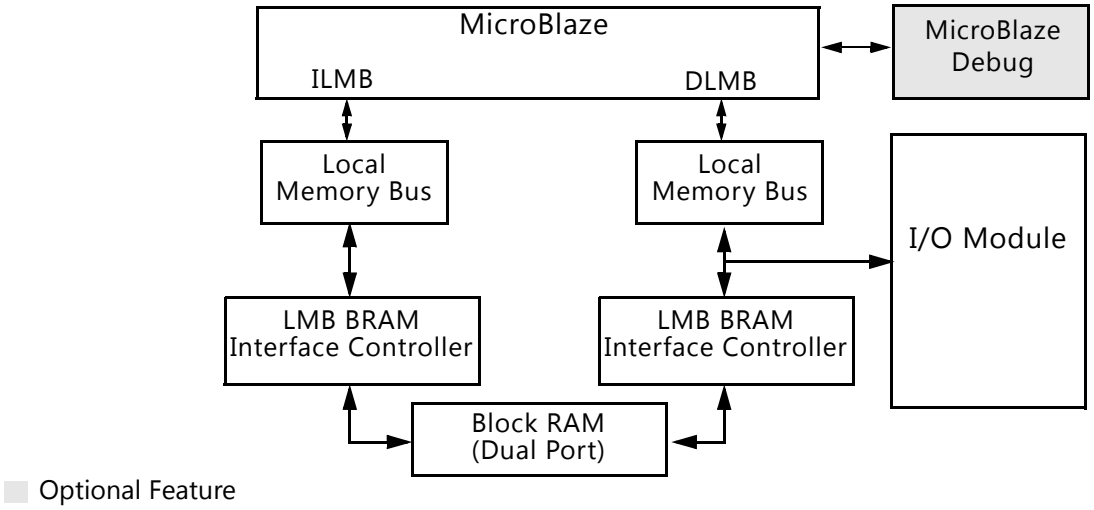

Figure 1-1: **MicroBlaze Micro Controller System**

Feature Summary

MicroBlaze

The MicroBlaze embedded processor soft core is a reduced instruction set computer (RISC) optimized for implementation in Xilinx® devices. Detailed information on the MicroBlaze processor can be found in the *MicroBlaze Processor Reference Guide* (UG081) [Ref 1].

The MicroBlaze parameters in the MicroBlaze MCS core are fixed except for the possibility to enable/disable the debug functionality, including debug UART. Table 4-2 shows the core parameter values. These values correspond to the MicroBlaze Configuration Wizard Minimum Area configuration.

Local Memory

Local memory is used for data and program storage and is implemented using block RAM. The size of the local memory is parameterized and can be between 4 kB and 128 kB. The local memory is connected to MicroBlaze through the Local Memory Bus, LMB, and the LMB BRAM Interface Controller cores. Detailed information on the LMB core can be found in the *Local Memory Bus (LMB) V10 Product Guide* (PG113) [Ref 2] and detailed information on the LMB BRAM Interface Controller core can be found in the *LMB BRAM Interface Controller Product Guide* (PG112) [Ref 3].

The memory sizes 4 kB, 8 kB, 16 kB, 32 kB, 64 kB and 128 kB require less resources, and should be used if possible.

The LMB Bus and the LMB BRAM Interface Controller core parameters are fixed except for the memory size. The parameter values can be found in Tables 4-4 to 4-7.

Debug

The MDM core connects MicroBlaze debug logic to the Xilinx Microprocessor Debugger (XMD). XMD can be used for downloading software, to set break points, view register and memory contents. If the Debug UART is enabled, XMD can also be utilized for standard input and standard output. Detailed information about MDM core can be found in the *MicroBlaze Debug Module (MDM) Product Guide* (PG115) [Ref 4].

The MDM parameters, except the JTAG user-defined register and the Debug UART, are fixed and their values can be found in Table 4-8.

When more than one MicroBlaze MCS core instance with debug enabled is included in the same design, a unique JTAG register must be used for each instance. When a single instance is used, the default value USER2 should be kept unchanged.

I/O Module

The I/O Module core is a light-weight implementation of a set of standard I/O functions commonly used in a MicroBlaze processor sub-system. Detailed information about the I/O Module core can be found in the *I/O Module Product Guide* (PG111) [Ref 5].

The I/O Module core registers are mapped at address 0x4000000, and the I/O Bus is mapped at address 0xC0000000-0xFFFFFFFF in the MicroBlaze memory space. The fixed I/O Module parameter values can be found in Table 4-3.

Licensing and Ordering Information

This Xilinx LogiCORE™ IP module is provided at no additional cost with the Xilinx Vivado® Design Suite under the terms of the Xilinx End User License. Information about this and other Xilinx LogiCORE IP modules is available at the Xilinx Intellectual Property page. For information about pricing and availability of other Xilinx LogiCORE IP modules and tools, contact your local Xilinx sales representative.

Product Specification

Standards

The I/O Bus interface provided by the I/O Module core is fully compatible with the Xilinx® Dynamic Reconfiguration Port (DRP). For a detailed description of the DRP, see the *7 Series FPGAs Configuration User Guide* (UG470) [Ref 6].

Performance

The frequency and latency of the modules in the MicroBlaze™ MCS core are optimized for use together with MicroBlaze. This means that the frequency targets are aligned to MicroBlaze targets as well as the access latency optimized for MicroBlaze data access.

Maximum Frequencies

For details about maximum frequencies, visit Performance and Resource Utilization.

Latency

Data read from I/O Module registers is available two clock cycles after the MicroBlaze load instruction is executed.

Data write to I/O Module registers is performed the clock cycle after the MicroBlaze store instruction is executed. Data accesses to peripherals connected on the I/O bus take three clock cycles plus the number of wait states introduced by the accessed peripheral.

Throughput

The maximum throughput when using the I/O bus is one read or write access every three clock cycles.

Resource Utilization

For details about resource utilization, visit Performance and Resource Utilization.

Port Descriptions

The I/O ports and signals for MicroBlaze MCS core are listed and described in Table 2-1.

Table 2-1: **MicroBlaze MCS Signals**

Port Name	MSB:LSB	I/O	Description
System Signals			
Clk		I	System clock
Reset		I	System reset
MicroBlaze Signals			
Trace_Valid_Instr		O	Valid instruction on trace port
Trace_Instruction	0:31	O	Instruction code
Trace_PC	0:31	O	Program counter
Trace_Reg_Write		O	Instruction writes to the register file
Trace_Reg_Addr	0:4	O	Destination register address
Trace_MSR_Reg	0:14	O	Machine status register
Trace_New_Reg_Value	0:31	O	Destination register update value
Trace_Jump_Taken		O	Branch instruction evaluated TRUE (taken)
Trace_Delay_Slot		O	Instruction is in delay slot of a taken branch
Trace_Data_AccessT		O	Valid D-side memory access
Trace_Data_Address	0:31	O	Address for D-side memory access
Trace_Data_Write_Value	0:31	O	Value for D-side memory write access
Trace_Data_Byte_Enable	0:3	O	Byte enables for D-side memory access
Trace_Data_Read		O	D-side memory access is a read
Trace_Data_Write		O	D-side memory access is a write
I/O Bus Signals			
IO_Addr_Strobe		O	Address strobe signals valid I/O Bus output signals
IO_Read_Strobe		O	I/O Bus access is a read
IO_Write_Strobe		O	I/O Bus access is a write
IO_Address	31:0	O	Address for access
IO_Byte_Enable	3:0	O	Byte enables for access
IO_Write_Data	31:0	O	Data to write for I/O Bus write access
IO_Read_Data	31:0	I	Read data for I/O Bus read access
IO_Ready		I	Ready handshake to end I/O Bus access

Table 2-1: **MicroBlaze MCS Signals *(Cont'd)***

Port Name	MSB:LSB	I/O	Description
UART Signals			
UART_Rx_IO		I	Receive Data
UART_Tx_IO		O	Transmit Data
UART_Interrupt		O	UART Interrupt
FIT Signals			
FITx_Interrupt(1)		O	FITx timer lapsed
FITx_Toggle(1)		O	Inverted FITx_Toggle when FITx timer lapses
PIT Signals			
PITx_Enable(1)		I	PITx count enable when C_PITx_PRESCALER = External
PITx_Interrupt(1)		O	PITx timer lapsed
PITx_Toggle(1)		O	Inverted PITx_Toggle when PITx lapses
GPO Signals			
GPOx(1)	[C_GPOx_SIZE - 1]:0	O	GPOx Output
GPI Signals			
GPIx(1)	[C_GPIx_SIZE - 1]:0	I	GPIx Input
GPIx_Interrupt(1)		O	GPIx input changed
INTC Signals			
INTC_Interrupt(2)	0:[C_INTC_INTR_SIZE - 1]	I	External interrupt inputs

Notes:

1. x = 1, 2, 3 or 4
2. Each of the interrupt inputs is treated as synchronous to the clock unless the corresponding bit in the parameter C_INTC_ASYNC_INTR is set. In that case, the input is synchronized with the number of flip-flops defined by the parameter C_INTC_NUM_SYNC_FF.

Register Space

The address map for the MicroBlaze MCS core is shown in Table 2-2.

Table 2-2: **MicroBlaze MCS Address Map**

Address (hex)	Name	Access Type	Description
0x0 - C_MEMSIZE-1	Local Memory	RW	Local Memory for MicroBlaze software
C_MEMSIZE - 0x7FFFFFFF	Reserved		
0x80000000 - 0x800000FF	I/O Module	RW	Mapped to I/O Module registers
0x80000100 - 0xBFFFFFFF	Reserved		
0xC0000000 - 0xFFFFFFFF	I/O Bus	RW	Mapped to I/O Bus address output

Designing with the Core

This chapter includes guidelines and additional information to facilitate designing with the core.

General Design Guidelines

I/O Module Interfaces

See the *I/O Module Product Guide* (PG111) [Ref 5] for design guidelines for the I/O Bus, UART, Fixed Interval Timer, Programmable Interval Timer, General Purpose Output, General Purpose Input, and Interrupt Controller. All of these interfaces are directly connected to the I/O Module inside the MicroBlaze™ MCS core.

MicroBlaze Trace Signals

See the *MicroBlaze Processor Reference Guide* (UG081) [Ref 1] for a detailed description of the MicroBlaze Trace signals. The Trace signals are directly connected to the MicroBlaze™ processor inside the MicroBlaze MCS core.

MicroBlaze Debug Module

See the *Xilinx SDK Help* [Ref 7] and the *MicroBlaze Debug Module (MDM) Product Guide* (PG115) [Ref 4] for a description of debugging with the MicroBlaze Debug Module (MDM) core.

Clocking

The MicroBlaze MCS core is fully synchronous with all clocked elements clocked with the `Clk` input.

Resets

The `Reset` input is the master reset input signal for the entire MicroBlaze MCS core. In addition, the entire MicroBlaze MCS core or only the MicroBlaze processor can be reset from the Xilinx Microprocessor Debugger (XMD), provided that debug is enabled.

Protocol Description

See the I/O bus timing diagrams in the *I/O Module Product Guide* (PG111) [Ref 5].

Design Flow Steps

This chapter describes customizing and generating the core, constraining the core, and the simulation, synthesis and implementation steps that are specific to this IP core. More detailed information about the standard Vivado® design flows and the IP integrator can be found in the following Vivado Design Suite user guides:

- *Vivado Design Suite User Guide: Designing IP Subsystems using IP Integrator* (UG994) [Ref 8]
- *Vivado Design Suite User Guide: Designing with IP* (UG896) [Ref 9]
- *Vivado Design Suite User Guide: Getting Started* (UG910) [Ref 10]
- *Vivado Design Suite User Guide: Logic Simulation* (UG900) [Ref 11]

Customizing and Generating the Core

This chapter includes information on using Xilinx tools to customize and generate the core using the Vivado Design Suite.

If you are customizing and generating the core in the Vivado IP integrator, see the *Vivado Design Suite User Guide: Designing IP Subsystems using IP Integrator* (UG994) [Ref 8] for detailed information. IP integrator might auto-compute certain configuration values when validating or generating the design. To check whether the values do change, see the description of the parameter in this chapter. To view the parameter value you can run the `validate_bd_design` command in the Tcl console.

You can customize the IP for use in your design by specifying values for the various parameters associated with the IP core using the following steps:

1. Select the IP from the IP catalog.
2. Double-click the selected IP or select the Customize IP command from the toolbar or right-click menu.

For details, see the *Vivado Design Suite User Guide: Designing with IP* (UG896) [Ref 9] and the *Vivado Design Suite User Guide: Getting Started* (UG910) [Ref 10].

Note: Figures in this chapter are illustrations of the MicroBlaze™ MCS core interface in the Vivado Integrated Design Environment (IDE). This layout might vary from the current version.

The MicroBlaze MCS core parameters are divided into eight tabs: Board, MCS, UART, FIT, PIT, GPO, GPI and Interrupts.

The Board tab is shown in Figure 4-1.

TIP: The board tab is only visible when a board has been defined for the project being used.

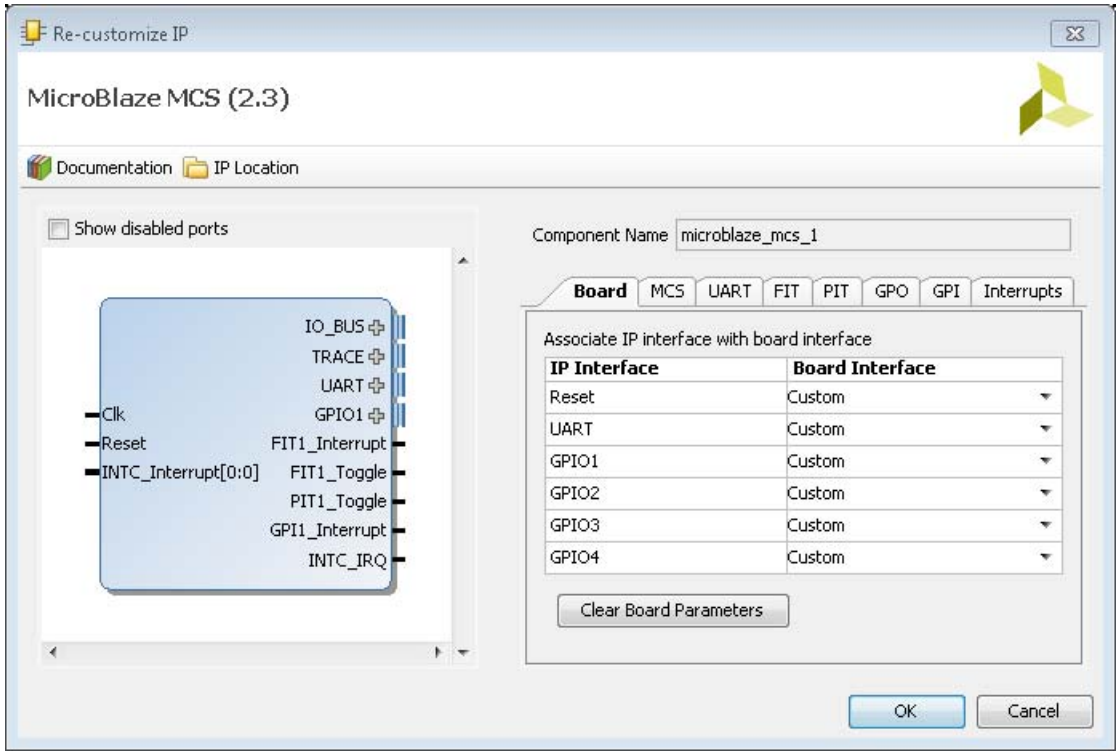

Figure 4-1: **Board Tab**

Generate Board Based IO Constraints - Enable board specific GPI, GPO, Reset, and UART interfaces. Board constraints are automatically generated for the selected interfaces.

Associate IP interface ... - Table to select board interface for Reset, UART, GPIO1, GPIO2, GPIO3, or GPIO4 interface.

The MCS parameter tab is shown in Figure 4-2.

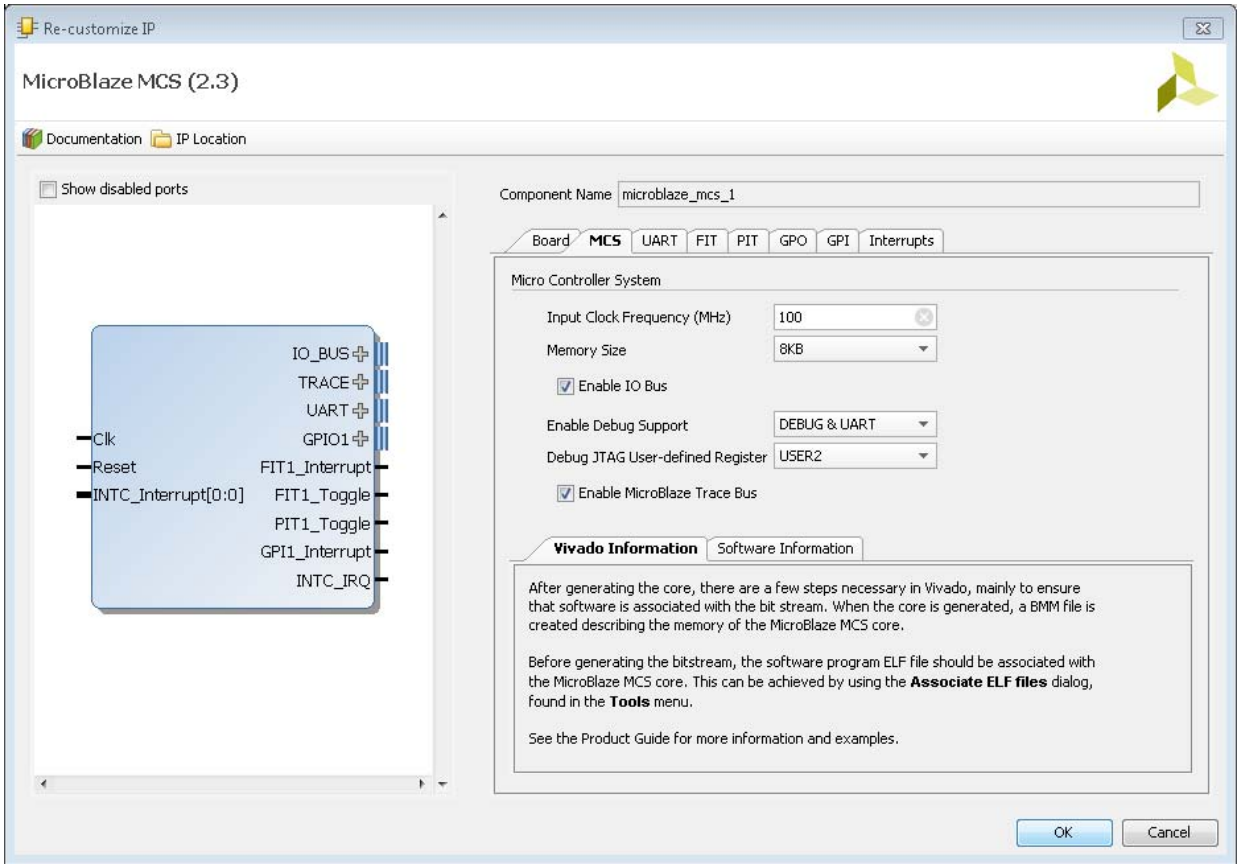

Figure 4-2: **MCS Parameter Tab**

- **Memory Size** - Defines the local memory size, used to store the MicroBlaze processor software program instructions and data. Increase this value if the software program does not fit in available memory.
- **Enable I/O Bus** - Enables I/O Bus port.
- **Enable Debug Support** - When debug support is enabled (DEBUG ONLY or DEBUG & UART), it is possible to debug the software using JTAG, from the Xilinx Software Development Kit (SDK) or directly using the Xilinx Microprocessor Debugger (XMD). When Debug UART is enabled (DEBUG & UART) it is also possible to use XMD for software program standard input and standard output.
- **Debug JTAG User-defined Register** - Specifies the JTAG user-defined register for debug. When more than one MicroBlaze MCS instance with debug enabled is included in the same design, a unique JTAG register must be used for each instance. When a single instance is used, the default value USER2 should be kept unchanged.
- **Enable MicroBlaze Trace Bus** - This option enables the MicroBlaze Trace bus, which provides access to several internal processor signals for trace purposes.

The UART parameter tab is shown in Figure 4-3.

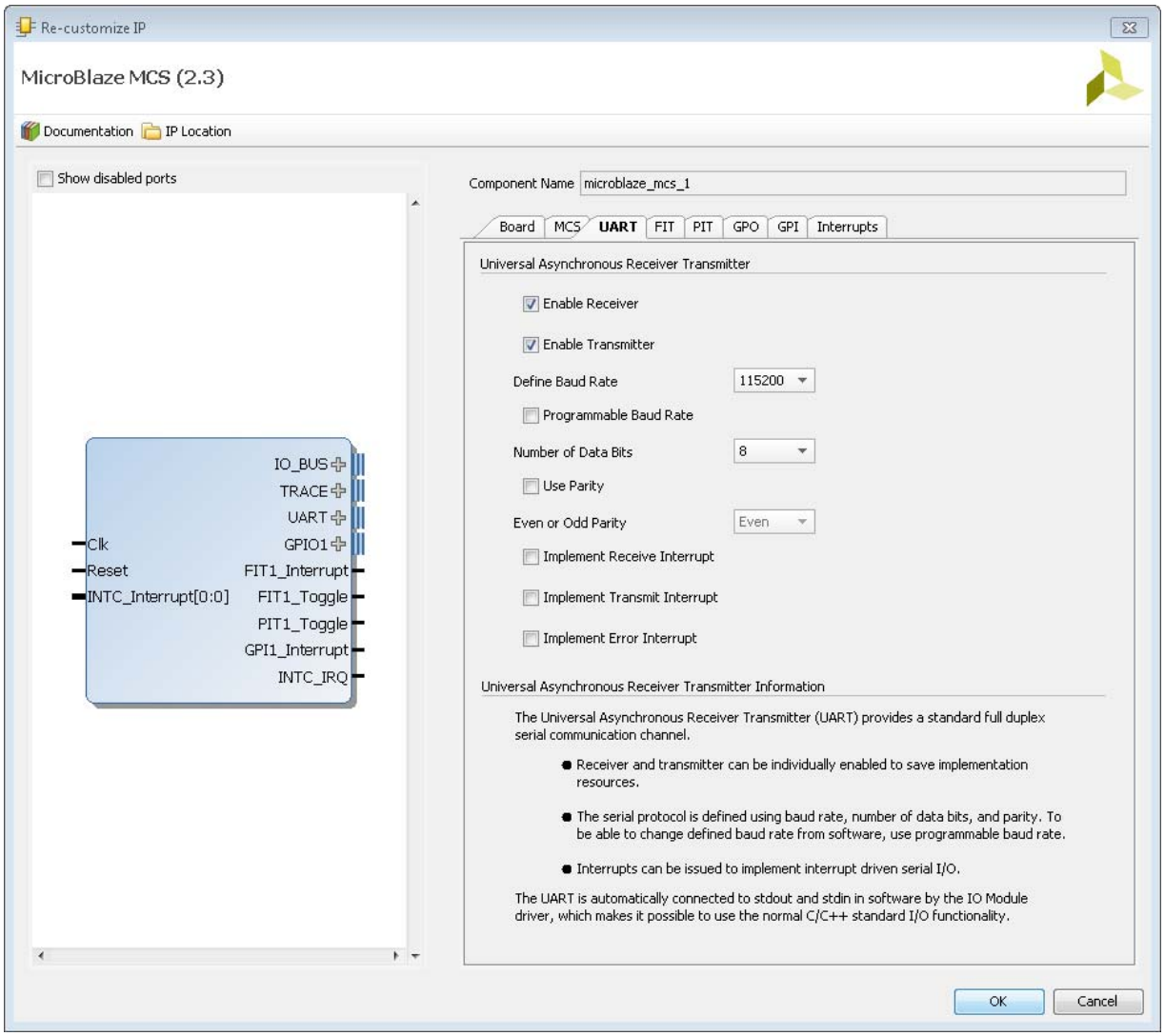

Figure 4-3: **UART Parameter Tab**

- **Enable Receiver** - Enables UART receiver for character input. This is automatically connected to standard input (`stdin`) in the software program.
- **Enable Transmitter** - Enables UART transmitter for character output. This is automatically connected to standard output (`stdout`) in the software program.
- **Define Baud Rate** - Sets the UART baud rate. To get the correct baud rate, the input clock frequency must also be correctly defined.
- **Programmable Baud Rate** - Determines if the UART baud rate is programmable. The default baud rate is calculated based on the input clock frequency and the defined baud rate.
- **Number of Data Bits** - Defines the number of data bits used by the UART. Should almost always be set to 8.

- **Use Parity** - Enable this parameter to use parity checking of the UART characters.
- **Even or Odd Parity** - Select odd or even parity. Only available when parity is used.
- **Implement Receive Interrupt** - Generate an interrupt when the UART has received a character. When the interrupt is not enabled the UART must be polled to check if data has been received.
- **Implement Transmit Interrupt** - Generate an interrupt when the UART has sent a character. When the interrupt is not enabled the UART must be polled to wait until data has been transmitted.
- **Implement Error Interrupt** - Generate an interrupt if an error occurs when the UART receives a character. This error can be a framing error, an overrun error or a parity error (if parity is used), When the interrupt is not enabled the UART must be polled to check if an error has occurred after a character has been received.

The FIT parameter tab showing the parameters for one of the four timers is illustrated in Figure 4-4.

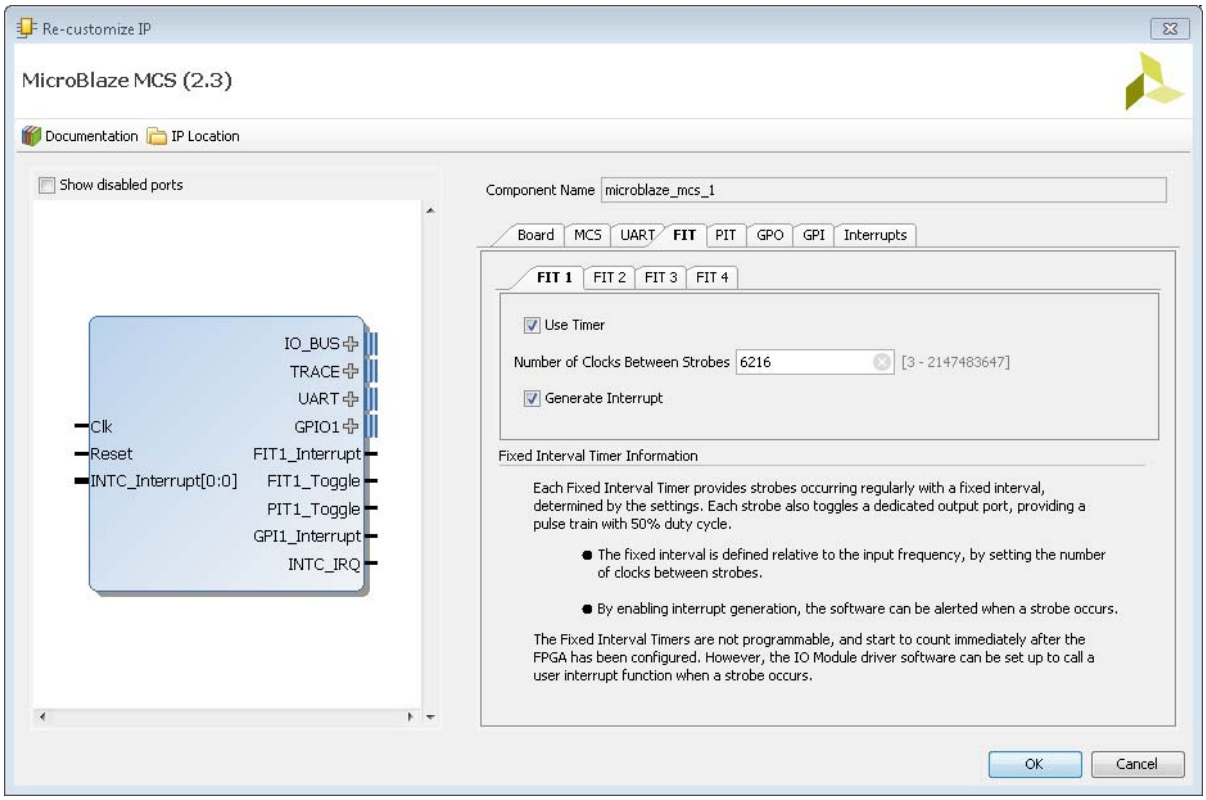

Figure 4-4: **FIT Parameter Tab**

- **Use FIT** - Enable the Fixed Interval Timer.
- **Number of Clocks Between Strobes** - The number of clock cycles between each strobe.
- **Generate Interrupt** - Generate an interrupt for each Fixed Interval Timer strobe.

The PIT parameter tab showing the parameters for one of the four timers is illustrated in Figure 4-5.

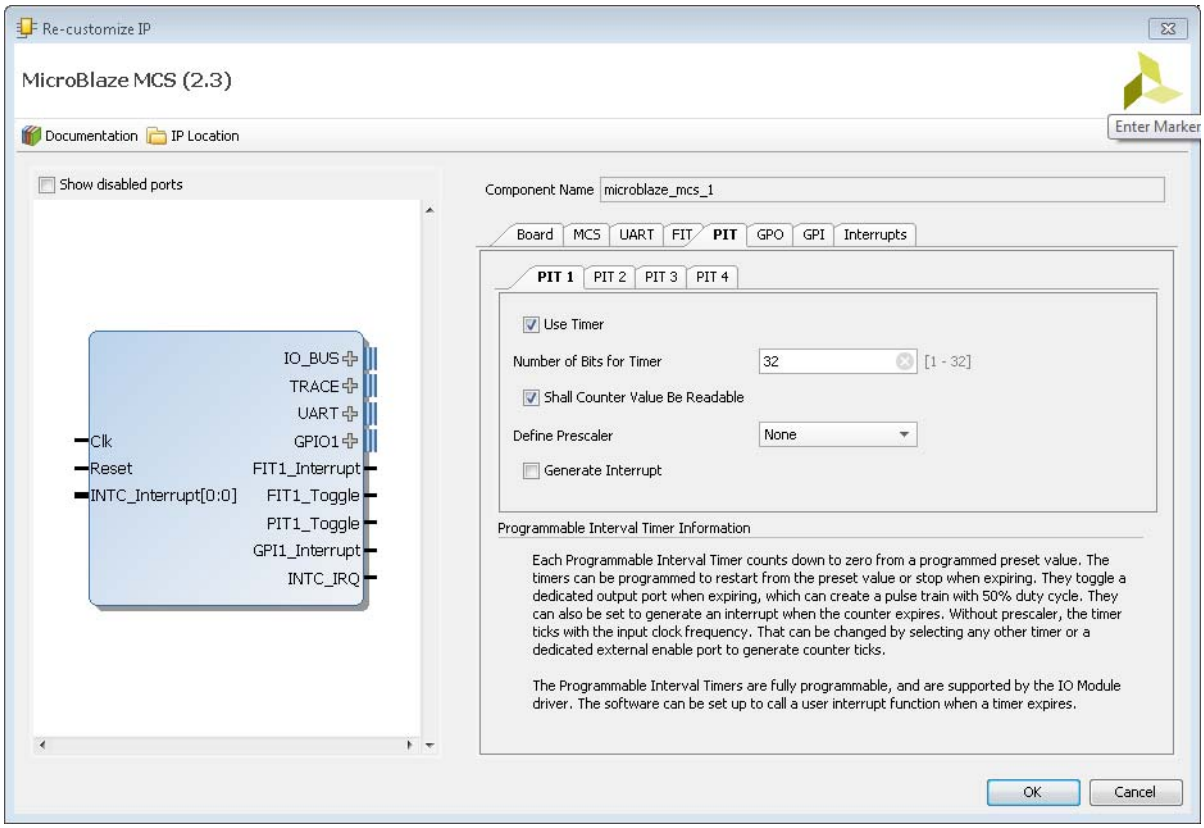

Figure 4-5: **PIT Parameter Tab**

- **Use PIT** - Enable the Programmable Interval Timer.
- **Number of Bits for Timer** - The maximum number of cycles to count before stopping or restarting.
- **Shall Counter Value be Readable** - The Programmable Interval Timer counter is readable by software when this parameter is set.

RECOMMENDED: It is recommended that you keep this enabled unless resource usage is critical.

- **Define Prescaler** - Selects a prescaler as source for the Programmable Interval Timer count. When no prescaler is selected the core input clock is used. Any Programmable Interval Timer or Fixed Interval Timer can be used as prescaler, as well as a dedicated external enable input.
- **Generate Interrupt** - Generate an interrupt when the Programmable Interval Timer has counted down to zero.

The GPO parameter tab showing the parameters for the four General Purpose Output ports is illustrated in Figure 4-6.

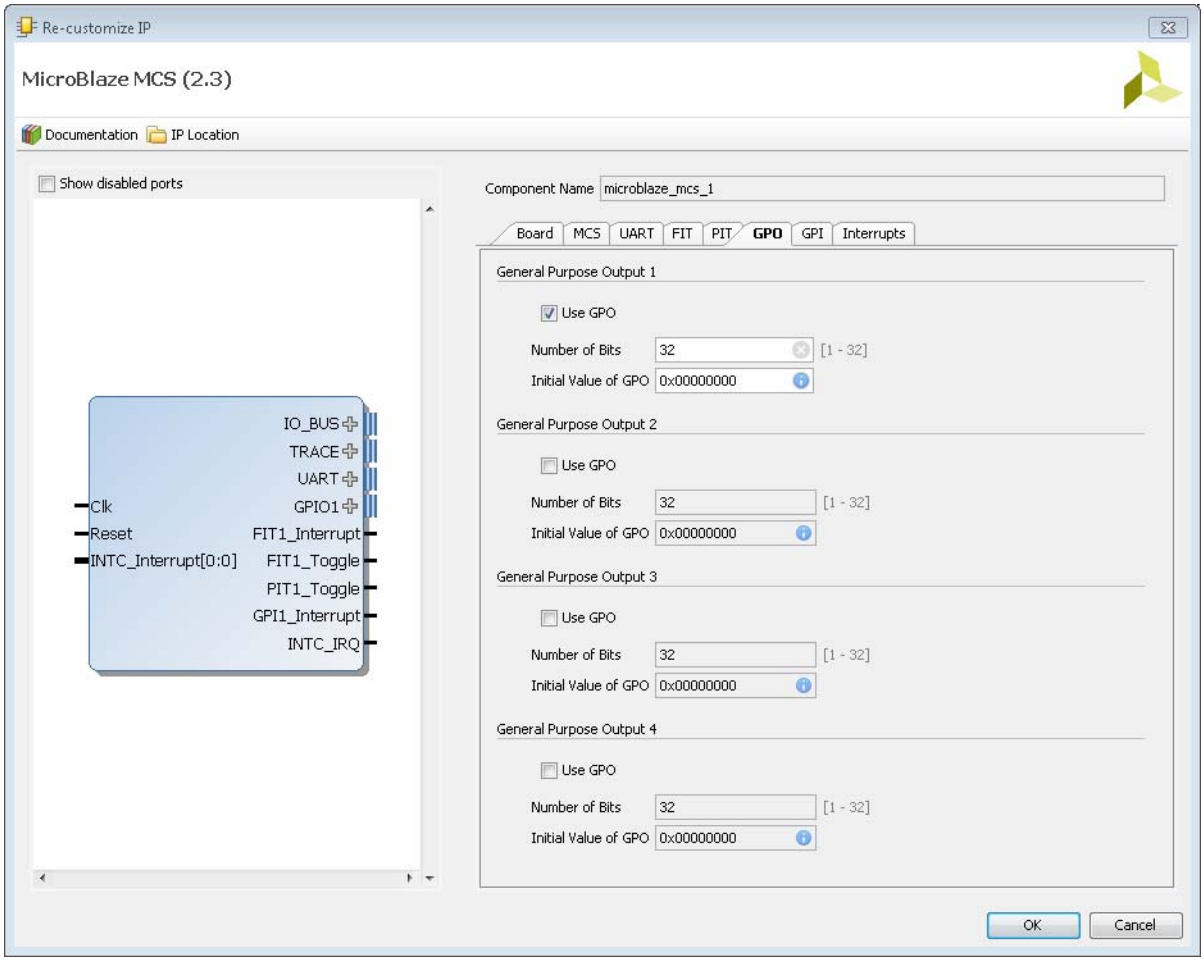

Figure 4-6: **GPO Parameter Tab**

- **Use GPO** - Enable the General Purpose Output port.
- **Number of Bits** - Set the number of bits of the General Purpose Output port.
- **Initial Value of GPO** - Set the initial value of the General Purpose Output port. The right most bit in the value is assigned to bit 0 of the port, the next right most to bit 1, and so on.

The GPI parameter tab showing the parameters for the four General Purpose Input ports is illustrated in Figure 4-7.

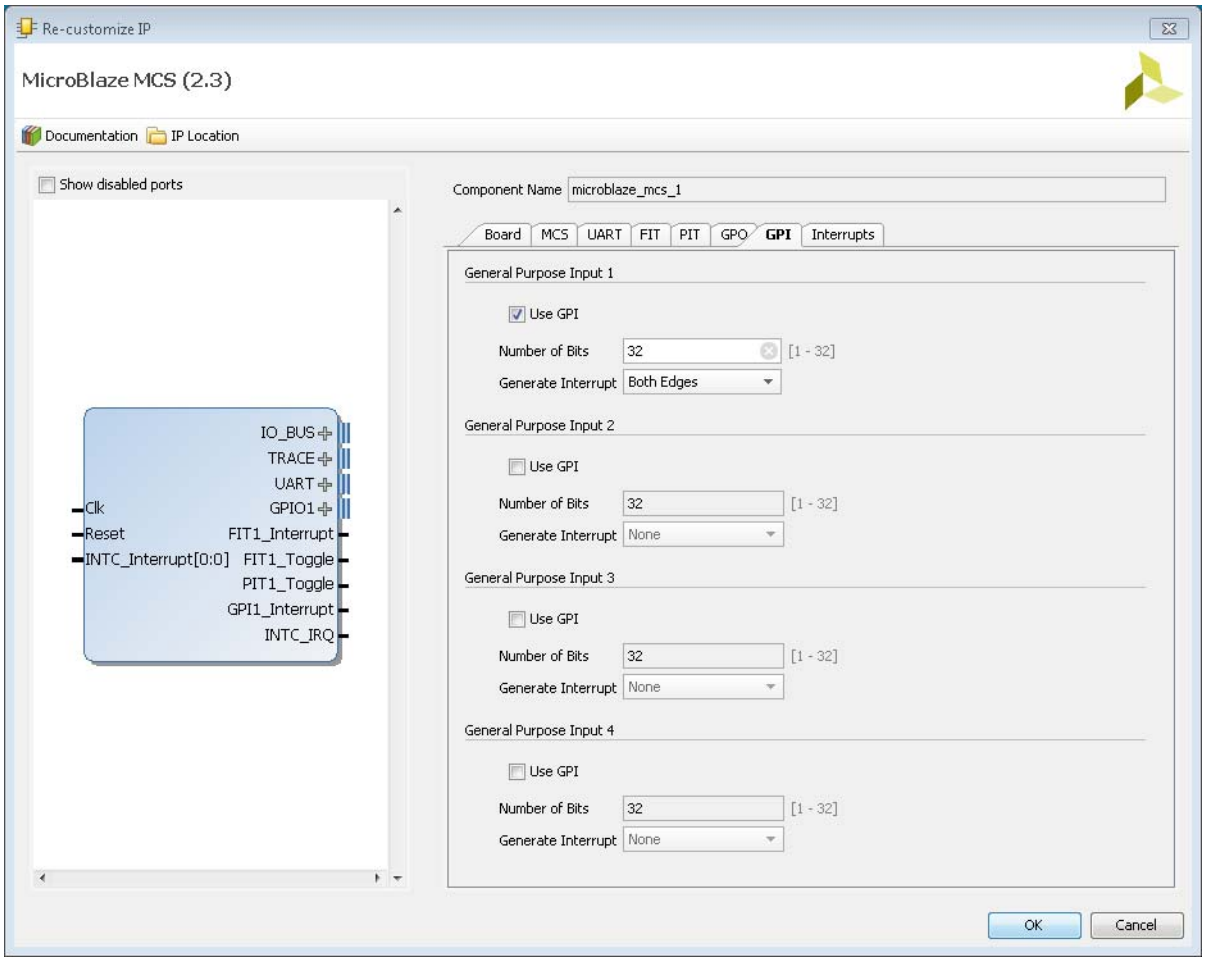

Figure 4-7: **GPI Parameter Tab**

- **Use GPI** - Enable the General Purpose Input port.
- **Number of Bits** - Set the number of bits of the General Purpose Input port.
- **Generate Interrupt** - Generate an interrupt when a General Purpose Input changes in the specified way - either any change (Both Edges), only when changed from 0 to 1 (Rising Edge), or only when changed from 1 to 0 (Falling Edge).

The Interrupts parameter tab is shown in Figure 4-8.

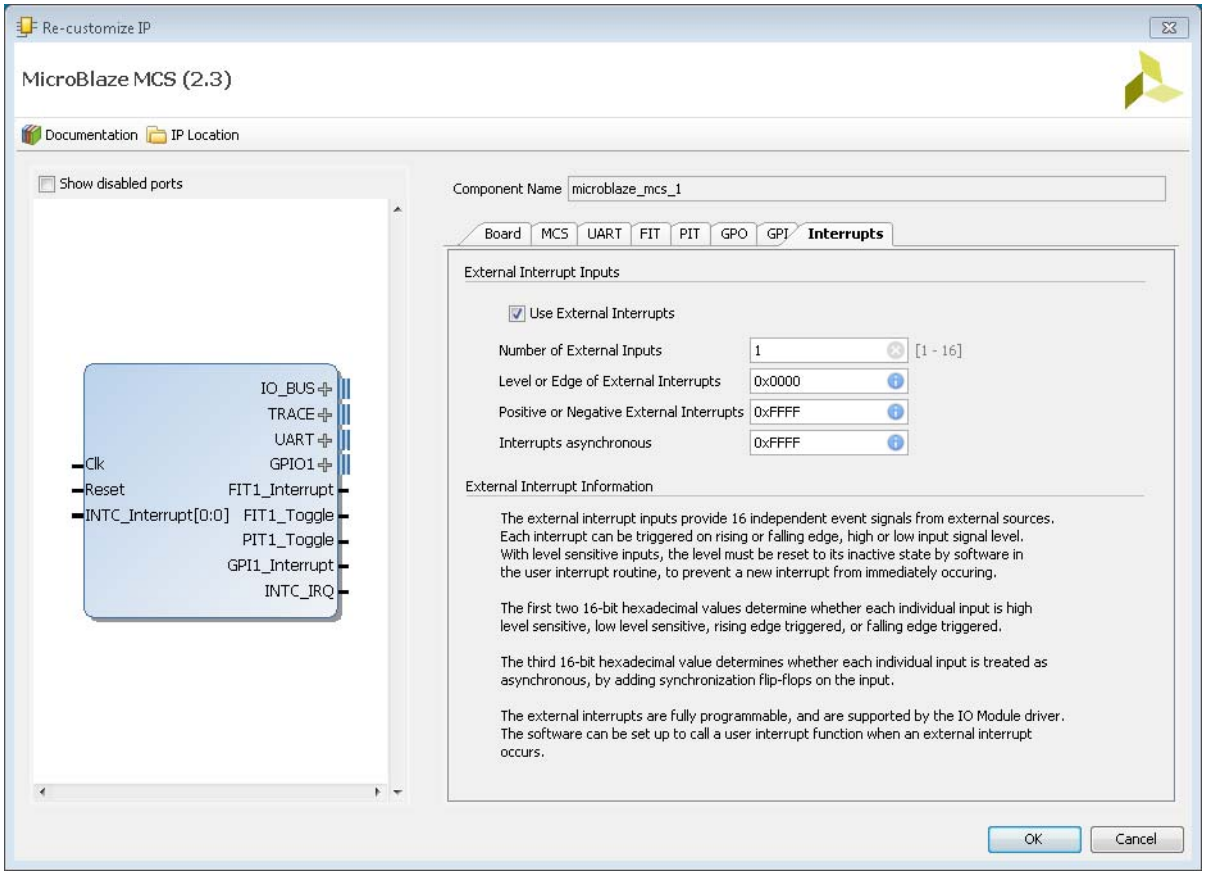

Figure 4-8: **Interrupts Parameter Tab**

- **Use External Interrupts** - Enable the use of external interrupt inputs.
- **Number of External Inputs** - Select the number of used external interrupt inputs.
- **Level or Edge of External Interrupts** - Select whether the input is considered level sensitive or edge triggered. Each bit in the value corresponds to the equivalent interrupt input. When a bit is set to one, the interrupt is edge triggered, otherwise it is level sensitive.
- **Positive or Negative External Interrupts** - Select whether to use High or Low level for level sensitive interrupts, and rising or falling edge for edge triggered interrupts. Each bit in the value corresponds to the equivalent interrupt input When a bit is set to one, High level or rising edge is used, otherwise Low level or falling edge is used.
- **Use Low-latency Interrupt Handling** - Enable the use of low-latency interrupt handling.
- **Interrupts Asynchronous** - Set whether to treat interrupts as asynchronous, by adding synchronization flip-flops on the input. Each bit in the value corresponds to the equivalent interrupt input When a bit is set to one, the input is treated as asynchronous.

Parameter Values

To create a MicroBlaze MCS core that is uniquely tailored for a specific system, certain features can be parameterized. This makes it possible for you to configure a component that only uses the resources required by the system, and operates with the best possible performance. The features that can be parameterized in the MicroBlaze MCS core are shown in Table 4-1.

The internal modules of the MicroBlaze MCS core have fixed configurations detailed in:

- Table 4-2 - MicroBlaze
- Table 4-3 - I/O Module
- Table 4-4 and Table 4-5 - LMB v10
- Table 4-6 and Table 4-7 - LMB BRAM IF Controller
- Table 4-8 - MicroBlaze Debug Module

Table 4-1: **MicroBlaze MCS Parameters**

Parameter Name	Feature/Description	Allowable Values	Default Value	VHDL Type
MCS Parameters				
C_FAMILY(1)	FPGA architecture	Supported architectures	kintex7	string
C_XDEVICE(1)	Device name	Supported devices	xc7k325t	string
C_XPACKAGE(1)	FPGA package name	Supported packages	ffg900	string
C_XSPEEDGRADE(1)	FPGA speed grade	Supported speed grades	-2	string
C_MICROBLAZE_INSTANCE(1)	Instance name		microblaze_0	string
C_PATH	Hierarchical path from top of design to MCS core instance		mb/UO	
C_FREQ	Frequency of CLK input		100000000	integer
C_MEMSIZE	Local memory size in bytes	4096 = 4KB 8192 = 8KB 12288 = 12KB 16384 = 16KB 20480 = 20KB 24576 = 24KB 32768 = 32KB 36864 = 36KB 40960 = 40KB 49152 = 48KB 65536 = 64KB 69632 = 68KB 73728 = 72KB 81920 = 80KB 98304 = 96KB 131072=128KB	8192	integer
C_DEBUG_ENABLED	Enable implementation of debug	0 = NONE 1 = DEBUG ONLY 2 = DEBUG & UART	0	Integer
C_JTAG_CHAIN	Select JTAG user-defined register	1 = USER1 2 = USER2 3 = USER3 4 = USER4	2	Integer

Table 4-1: **MicroBlaze MCS Parameters *(Cont'd)***

Parameter Name	Feature/Description	Allowable Values	Default Value	VHDL Type
I/O Bus Parameter				
C_USE_IO_BUS	Use I/O Bus	0 = Not Used 1 = Used	0	integer
UART Parameters				
C_USE_UART_RX	Use UART Receive	0 = Not Used 1 = Used	0	integer
C_USE_UART_TX	Use UART Transmit	0 = Not Used 1 = Used	0	integer
C_UART_BAUDRATE	Baud rate of the UART in bits per second	integer (for example 115200)	9600	integer
C_UART_PROG_ BAUDRATE	Programmable UART baud rate	0 = Not Used 1 = Used	0	integer
C_UART_DATA_BITS	The number of data bits in the serial frame	5 - 8	8	integer
C_UART_USE_PARITY	Determines whether parity is used or not	0 = No Parity 1 = Use Parity	0	integer
C_UART_ODD_PARITY	If parity is used, determines whether parity is odd or even	0 = Even Parity 1 = Odd Parity	0	integer
C_UART_RX_INTERRUPT	Use UART RX Interrupt in INTC	0 = Not Used 1 = Used	0	integer
C_UART_TX_INTERRUPT	Use UART TX Interrupt in INTC	0 = Not Used 1 = Used	0	integer
C_UART_ERROR_ INTERRUPT	Use UART ERROR Interrupt in INTC	0 = Not Used 1 = Used	0	integer
FIT Parameters				
C_USE_FITx(1)	Enable implementation of FIT	0 = Not Used 1 = Used	0	integer
C_FITx_No_CLOCKS(2)	The number of clock cycles between strobes	>2	6216	integer
C_FITx_INTERRUPT(2)	Use FITx_Interrupt in INTC	0 = Not Used 1 = Used	0	integer
PIT Parameters				
C_USE_PITx(2)	Enable implementation of PIT	0 = Not Used 1 = Used	0	integer
C_PITx_SIZE(2)	Size of PITx counter	1 - 32	1	integer
C_PITx_READABLE(2)	Make PITx counter software readable	0 = Not SW readable 1 = SW readable	1	integer

Table 4-1: **MicroBlaze MCS Parameters *(Cont'd)***

Parameter Name	Feature/Description	Allowable Values	Default Value	VHDL Type
C_PITx_PRESCALER[2][3]	Select PITx prescaler	0 = No prescaler 1 = FIT1 2 = FIT2 3 = FIT3 4 = FIT4 5 = PIT1 6 = PIT2 7 = PIT3 8 = PIT4 9 = External	0	integer
C_PITx_INTERRUPT[2]	Use PITx_Interrupt in INTC	0 = Not Used 1 = Used	0	integer
GPO Parameters				
C_USE_GPOx[2]	Use GPOx	0 = Not Used 1 = Used	0	integer
C_GPOx_SIZE[2]	Size of GPOx	1 - 32	32	integer
C_GPOx_INIT[2]	Initial value for GPOx	Fit Range (31:0)	all zeros	std_logic_vector
GPI Parameters				
C_USE_GPIx[2]	Use GPIx	0 = Not Used 1 = Used	0	integer
C_GPIx_SIZE[2]	Size of GPIx	1 - 32	32	integer
C_GPIx_INTERRUPT[2]	Use GPIx_Interrupt in INTC	0 = None 1 = Both Edges 2 = Rising Edge 3 = Falling Edge	0	integer
INTC Parameters				
C_INTC_USE_EXT_INTR	Use I/O Module external interrupt inputs	0 = Not Used 1 = Used	0	integer
C_INTC_INTR_SIZE	Number of external interrupt inputs used	1 - 16	1	integer
C_INTC_LEVEL_EDGE	Level or edge triggered for each external interrupt	For each bit: 0 = Level 1 = Edge	level	std_logic_vector
C_INTC_POSITIVE	Polarity for each external interrupt	For each bit: 0 = active-Low 1 = active-High	active-High	std_logic_vector
C_INTC_ASYNC_INTR	Asynchronous interrupt for each external interrupt	For each bit: 0 = Synchronous 1 = Asynchronous	Asynchronous	std_logic_vector
C_INTC_NUM_SYNC_FF[1]	Number of synchronization flip-flops	0 - 7	2	integer

Notes:

1. Values automatically populated by tool.
2. x=1, 2, 3 or 4.
3. Selecting PIT prescaler the same as PITx is illegal; for example, PIT2 cannot be prescaler to itself.

Table 4-2: **Internal MicroBlaze Parameters Settings**

Parameter Name	Feature/Description	Value
C_FAMILY	Target family	Value of MicroBlaze MCS parameter C_FAMILY
C_AREA_OPTIMIZED	Select implementation to optimize area with lower instruction throughput	1
C_INTERCONNECT	Select interconnect 2= AXI	2
C_ENDIANNESS	Select endianness (1 = Little endian)	1
C_FAULT_TOLERANT	Implement fault tolerance	0
C_LOCKSTEP_SLAVE	Lockstep Slave	0
C_AVOID_PRIMITIVES	Disallow FPGA primitives	0
C_PVR	Processor version register mode selection All other PVR parameters are don't care.	0
C_RESET_MSR	Reset value for MSR register	0x00
C_INSTANCE	Instance Name	Value of MicroBlaze MCS parameter C_MICROBLAZE_INSTANCE
C_D_AXI	Data side AXI interface All other data side AXI parameters are don't care.	0
C_D_LMB	Data side LMB interface	1
C_I_AXI	Instruction side AXI interface. All other instruction side AXI parameters are don't care.	0
C_I_LMB	Instruction side LMB interface	1
C_USE_BARREL	Include barrel shifter	0
C_USE_DIV	Include hardware divider	0
C_USE_HW_MUL	Include hardware multiplier	0
C_USE_FPU	Include hardware floating point unit	0
C_USE_MSR_INSTR	Enable use of instructions: MSRSET and MSRCLR	0
C_USE_PCMP_INSTR	Enable use of instructions: CLZ, PCMPBF, PCMPEQ, and PCMPNE	0
C_USE_REORDER_INSTR	Enable use of instructions: LBUR, LHUR, LWR, SBR,SHR, SWR, SWAPB, and SWAPH	0
C_*EXCEPTION*(1) C_OPCODE_0x0_ILLEGAL C_USE_STACK_PROTECTION	No exceptions are used	0
C_DEBUG_ENABLED	MDM Debug interface	Value of MicroBlaze MCS parameter C_DEBUG_ENABLED

Table 4-2: **Internal MicroBlaze Parameters Settings *(Cont'd)***

Parameter Name	Feature/Description	Value
C_NUMBER_OF_PC_BRK	Number of hardware breakpoints	Value of MicroBlaze MCS parameter C_DEBUG_ENABLED
C_NUMBER_OF_RD_ADDR_BRK	Number of read address watchpoints	0
C_NUMBER_OF_WR_ADDR_BRK	Number of write address watchpoints	0
C_INTERRUPT_IS_EDGE	Level/Edge interrupt	0
C_EDGE_IS_POSITIVE	Negative/positive edge interrupt	1
C_FSL_LINKS	Number of AXI stream interfaces. All other stream parameters are don't care	0
C_USE_ICACHE	Instruction cache. All other instruction cache parameters are don't care	0
C_USE_DCACHE	Data cache. All other data cache parameters are don't care	0
C_USE_MMU	Memory management. All other MMU parameters are don't care	0
C_USE_INTERRUPT	Enable interrupt handling	2
C_USE_EXT_BRK	Enable external break handling	Value of MicroBlaze MCS parameter C_DEBUG_ENABLED
C_USE_EXT_NM_BRK	Enable external non-maskable break handling	Value of MicroBlaze MCS parameter C_DEBUG_ENABLED
C_USE_BRANCH_TARGET_CACHE	Enable branch target cache. All other BTC parameters are don't care	0

Notes:

1. * denotes wildcard and represents any number of characters or numbers.

Table 4-3: **Internal I/O Module Parameters Settings**

Parameter Name	Feature/Description	Value
C_BASEADDR	LMB I/O Module register base address	0x80000000
C_HIGHADDR	LMB I/O Module register high address	0x8000FFFF
C_MASK	LMB I/O Module register address space decode mask	0xC0000000
C_IO_HIGHADDR	LMB I/O Module I/O bus base address	0xC0000000
C_IO_LOWADDR	LMB I/O Module I/O bus address	0xFFFFFFFF
C_IO_MASK	LMB I/O Module I/O bus address space decode mask	0xC0000000
C_LMB_AWIDTH	LMB address bus width	32
C_LMB_DWIDTH	LMB data bus width	32
C_INTC_HAS_FAST	Use fast interrupt mode	1
C_INTC_ADDR_WIDTH	Interrupt address width	12 - 16(1)

Notes:

1. Value depends on C_MEMSIZE: 12 for 4096, 13 for 8192, 14 for 16384, 15 for 32768, and 16 for 65536.

Table 4-4: **Internal LMB_v10 Parameters Settings (ILMB)**

Parameter Name	Feature/Description	Value
C_LMB_NUM_SLAVES	Number of LMB slaves	1
C_LMB_AWIDTH	LMB address bus width	32
C_LMB_DWIDTH	LMB data bus width	32
C_EXT_RESET_HIGH	Level of external reset	1 = active-High reset

Table 4-5: **Internal LMB_v10 Parameters Settings (DLMB)**

Parameter Name	Feature/Description	Value
C_LMB_NUM_SLAVES	Number of LMB slaves	2
C_LMB_AWIDTH	LMB address bus width	32
C_LMB_DWIDTH	LMB data bus width	32
C_EXT_RESET_HIGH	Level of external reset	1 = active-High reset

Table 4-6: **Internal LMB BRAM IF Controller Parameters Settings (ILMB Controller)**

Parameter Name	Feature/Description	Value
C_BASEADDR	LMB BRAM base address	0
C_HIGHADDR	LMB BRAM high address	Value of MicroBlaze MCS Parameter C_MEMSIZE
C_MASK	LMB decode mask	0x80000000
C_LMB_AWIDTH	LMB address bus width	32
C_LMB_DWIDTH	LMB data bus width	32
C_ECC	Implement error correction and detection All other ECC as well AXI parameters are don't care	0 = No ECC

Table 4-7: **Internal LMB BRAM IF Controller Parameters Settings (DLMB Controller)**

Parameter Name	Feature/Description	Value
C_BASEADDR	LMB BRAM base address	0
C_HIGHADDR	LMB BRAM high address	Value of MicroBlaze MCS Parameter C_MEMSIZE
C_MASK	LMB decode mask	0x80000000
C_LMB_AWIDTH	LMB address bus width	32
C_LMB_DWIDTH	LMB data bus width	32
C_ECC	Implement error correction and detection All other ECC as well as AXI parameters are don't care	0 = No ECC

Table 4-8: **MicroBlaze Debug Module Parameters Settings**

Parameter Name	Feature/Description	Value
C_FAMILY	FPGA architecture	Value of MicroBlaze MCS Parameter C_FAMILY
C_MB_DBG_PORTS	Number of MicroBlaze debug ports	1
C_USE_UART	Enables the UART interface.	Set if MicroBlaze MCS Parameter C_DEBUG_ENABLED=2
C_DBG_MEM_ACCESS	Enable AXI memory access from debug	0
C_DBG_REG_ACCESS	Enable debug register access from AXI	0
C_USE_CROSS_TRIGGER	Enable cross trigger	0

Parameter - Port Dependencies

The width of many of the MicroBlaze MCS signals depends on design parameters. The dependencies between the design parameters and I/O signals are shown in Table 4-9.

Table 4-9: **Parameter-Port Dependencies**

Parameter Name	Ports (Port width depends on parameter)
C_INTC_INTR_SIZE	INTC_Interrupt
C_GPO1_SIZE	GPO1
C_GPO2_SIZE	GPO2
C_GPO3_SIZE	GPO3
C_GPO4_SIZE	GPO4
C_GPI1_SIZE	GPI1
C_GPI2_SIZE	GPI2
C_GPI3_SIZE	GPI3
C_GPI4_SIZE	GPI4

Tool Flow

The MicroBlaze MCS core uses the generic tool flow of all Vivado IP cores. The SDK software development flow is briefly described here.

Generic Vivado Tool Flow

The generic tool flow in Vivado is shown in Figure 4-9.

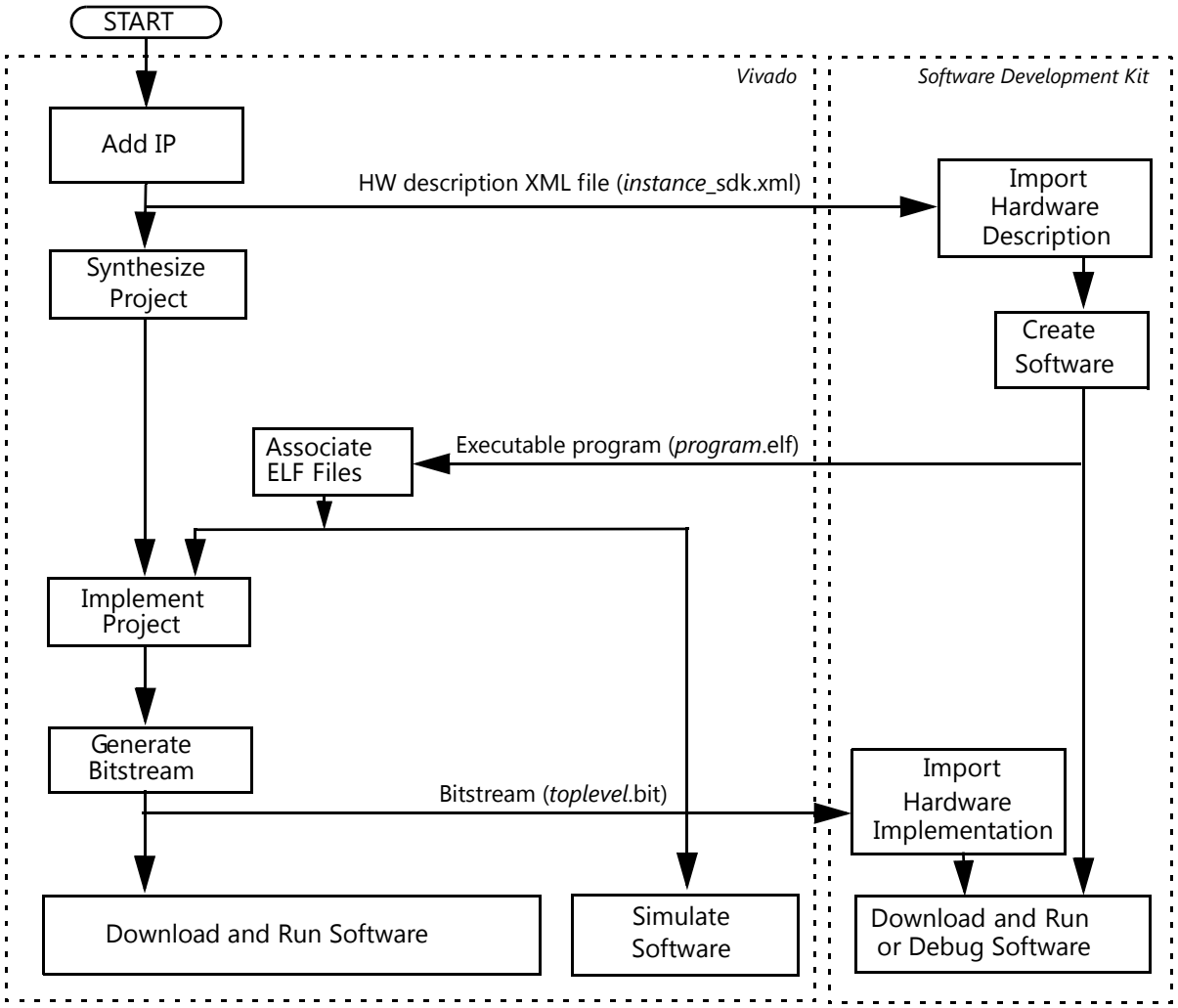

Figure 4-9: **Generic Vivado Tool Flow**

This flow shows the specific steps required to implement a project with the MicroBlaze MCS core in Vivado, and the relationship between the hardware and software tools.

- **Associate ELF Files:**

  This is the only manual step in Vivado, performed by selecting **Tools > Associate ELF Files...** in the menu. Initially, the default infinite loop ELF file, `mb_bootloop_le.elf`, is associated with the MicroBlaze MCS core. ELF files for implementation and simulation are specified separately.

  Note: The associated ELF files are imported into the project. This means that they are unaffected by changes during software development in SDK, but need to be re-imported to apply any changes.

The final bitstream updated with software is named download.bit, and is normally located in the project directory `project-name.runs/impl_1`. For additional information, see the *Vivado Design Suite User Guide: Designing with IP* (UG896) [Ref 9].

SDK

The SDK commands to achieve the MicroBlaze MCS specific steps above are detailed here:

- Import Hardware Description - For each MicroBlaze MCS component to import:
  - Select **File > New > Project...** in the menu.
  - Expand **Xilinx**, and select **Hardware Platform Specification**.
  - Click **Next**.
  - Click **Browse**, and navigate to the hardware description file which is typically called `project-name.srcs/sources_1/ip/component-name/component-name_sdk.xml`.
  - Click **Finish** to perform the import.

  After the hardware description has been imported, a standalone board support package can be created, which provides MicroBlaze processor-specific code, and the I/O Module software driver. The MicroBlaze MCS configuration is available in the generated file `microblaze_0/include/xparameters.h`.
- Import Hardware Implementation:
  - Select **Xilinx Tools > Program FPGA** in the menu.
  - Click the first **Browse** button, and navigate to the bitstream which is typically called `project-name.runs/impl_1/toplevel.bit`.
  - Click **Program** to perform the import and program the FPGA.

  Note: Including a BMM file updated with block RAM placement by using the second **Browse** button is not necessary, because SDK does not use this file. To update the bitstream with an ELF file when performing FPGA programming, SDK requires that an HDF file is used to import the hardware description and implementation, which is not provided with the MicroBlaze MCS Vivado flow.

For additional information, see the *Xilinx SDK Help* [Ref 7].

Output Generation

For details, see the *Vivado Design Suite User Guide: Designing with IP* (UG896) [Ref 9].

Constraining the Core

This section contains information about constraining the core in the Vivado Design Suite.

Required Constraints

This section is not applicable for this IP core.

Device, Package, and Speed Grade Selections

This section is not applicable for this IP core.

Clock Frequencies

This section is not applicable for this IP core.

Clock Management

The MicroBlaze MCS core is fully synchronous with all clocked elements clocked by the `Clk` input.

Clock Placement

This section is not applicable for this IP core.

Banking

This section is not applicable for this IP core.

Transceiver Placement

This section is not applicable for this IP core.

I/O Standard and Placement

This section is not applicable for this IP core.

Simulation

For comprehensive information about Vivado simulation components, as well as information about using supported third party tools, see the *Vivado Design Suite User*

Guide: Logic Simulation (UG900) [Ref 11].

IMPORTANT: *For cores targeting 7 series or Zynq®-7000 devices, UNIFAST libraries are not supported. Xilinx IP is tested and qualified with UNISIM libraries only.*

Synthesis and Implementation

For details about synthesis and implementation, see the *Vivado Design Suite User Guide: Designing with IP* (UG896) [Ref 9].

Migrating and Upgrading

This appendix contains information about migrating a design from ISE® to the Vivado® Design Suite, and for upgrading to a more recent version of the IP core. For customers upgrading in the Vivado Design Suite, important details (where applicable) about any port changes and other impact to user logic are included.

Migrating to the Vivado Design Suite

For information on migrating from Xilinx ISE Design Suite tools to the Vivado Design Suite, see the *ISE to Vivado Design Suite Migration Guide* (UG911) [Ref 12].

Upgrading in the Vivado Design Suite

This section provides information about any changes to the user logic or port designations that take place when you upgrade to a more current version of this IP core in the Vivado Design Suite.

The parameters, GPI1_INTERRUPT, GPI2_INTERRUPT, GPI3_INTERRUPT, and GPI4_INTERRUPT were changed in v2.0 of the core released in Vivado 2013.1, to add support for triggering GPI interrupts on either a rising or falling edge.

The parameters, USE_BOARD_FLOW, GPIO1_BOARD_INTERFACE, GPIO1_BOARD_INTERFACE, GPIO1_BOARD_INTERFACE, GPIO1_BOARD_INTERFACE, and UART_BOARD_INTERFACE were added in v2.0 of the core released in 2013.3, to add support for board level constraints.

There have been no port changes.

Debugging

This appendix includes details about resources available on the Xilinx Support website and debugging tools.

Finding Help on Xilinx.com

To help in the design and debug process when using the MicroBlaze MCS core, the Xilinx Support web page contains key resources such as product documentation, release notes, answer records, information about known issues, and links for obtaining further product support.

Documentation

This product guide is the main document associated with the MicroBlaze MCS core. This guide, along with documentation related to all products that aid in the design process, can be found on the Xilinx Support web page or by using the Xilinx® Documentation Navigator.

Download the Xilinx Documentation Navigator from the Design Tools tab on the Downloads page. For more information about this tool and the features available, open the online help after installation.

Answer Records

Answer Records include information about commonly encountered problems, helpful information on how to resolve these problems, and any known issues with a Xilinx product. Answer Records are created and maintained daily ensuring that users have access to the most accurate information available.

Answer Records for this core can be located by using the Search Support box on the main Xilinx support web page. To maximize your search results, use proper keywords such as

- Product name
- Tool message(s)
- Summary of the issue encountered

A filter search is available after results are returned to further target the results.

Master Answer Record for the MicroBlaze MCS Core

AR: 54414

Contacting Technical Support

Xilinx provides technical support at the Xilinx Support web page for this LogiCORE™ IP product when used as described in the product documentation. Xilinx cannot guarantee timing, functionality, or support of product if implemented in devices that are not defined in the documentation, if customized beyond that allowed in the product documentation, or if changes are made to any section of the design labeled DO NOT MODIFY.

To contact Xilinx Technical Support:

1. Navigate to the Xilinx Support web page.
2. Open a WebCase by selecting the WebCase link located under Additional Resources.

When opening a WebCase, include:

- Target FPGA including package and speed grade.
- All applicable Xilinx Design Tools and simulator software versions.
- Additional files based on the specific issue might also be required. See the relevant sections in this debug guide for guidelines about which file(s) to include with the WebCase.

Note: Access to WebCase is not available in all cases. Log in to the WebCase tool to see your specific support options.

Debug Tools

The main tool available to address MicroBlaze MCS design issues is the Vivado® Design Suite debug feature.

Vivado Design Suite Debug Feature

The Vivado Design Suite debug feature inserts logic analyzer and virtual I/O cores directly into your design. The debug feature also allows you to set trigger conditions to capture application and integrated block port signals in hardware. Captured signals can then be analyzed. This feature in the Vivado IDE is used for logic debugging and validation of a design running in Xilinx devices.

The Vivado logic analyzer is used with the logic debug IP cores, including:

- ILA 2.0 (and later versions)
- VIO 2.0 (and later versions)

See *Vivado Design Suite User Guide: Programming and Debugging* (UG908) [Ref 13].

Reference Boards

All 7 series Xilinx development boards support the MicroBlaze MCS core. These boards can be used to prototype designs and establish that the core can communicate with the system.

Simulation Debug

The simulation debug flow for Mentor Graphics Quests Simulator (QuestaSim) is described below. A similar approach can be used with other simulators.

- Check for the latest supported versions of QuestaSim in the Xilinx Design Tools: Release Notes Guide. Is this version being used? If not, update to this version.
- If using Verilog, do you have a mixed mode simulation license? If not, obtain a mixed-mode license.
- Ensure that the proper libraries are compiled and mapped. In the Vivado Design Suite **Flow > Simulation Settings** can be used to define the libraries.
- Have you associated the intended software program for the MicroBlaze processor with the simulation? Use the command **Tools > Associate ELF Files** in the Vivado Design Suite.

Hardware Debug

This section provides debug steps for common issues. The Vivado Design Suite debug feature is a valuable resource to use in hardware debug.

General Checks

Ensure that all the timing constraints for the core were properly incorporated from the example design and that all constraints were met during implementation.

- Does it work in post-place and route timing simulation? If problems are seen in hardware but not in timing simulation, this could indicate a PCB issue. Ensure that all clock sources are active and clean.
- If using MMCMs in the design, ensure that all MMCMs have obtained lock by monitoring the `locked` port.

Application Software Development

Xilinx Software Development Kit

The MicroBlaze™ MCS core can be used with the Xilinx® Software Development Kit (SDK), in the same way as any embedded system.

The specific steps needed with the MicroBlaze MCS core are described in SDK in Chapter 4.

Device Drivers

The I/O Module is supported by the I/O Module driver (*iomodule*), included with the Xilinx Software Development Kit. The I/O Module driver API is designed to be as similar as possible to the equivalent discrete peripheral driver API.

The correspondence between the *iomodule* driver API and the *intc*, *uartlite*, *tmrctr* and *gpio* driver API is listed in Table C-1. The I/O Module functions are equivalent to the discrete driver counterparts in terms of semantics and syntax, except that all take an `XIOModule` instance pointer.

Table C-1: **I/O Module Driver API Correspondence**

I/O Module Function	Discrete Function	Remark
XIOModule_Initialize	XIntc_Initialize	Should only be called once for the entire I/O Module driver
XIOModule_Start	XIntc_Start	
XIOModule_Stop	XIntc_Stop	
XIOModule_Connect	XIntc_Connect	
XIOModule_Disconnect	XIntc_Disconnect	
XIOModule_Enable	XIntc_Enable	
XIOModule_Disable	XIntc_Disable	
XIOModule_Acknowledge	XIntc_Acknowledge	
XIOModule_LookupConfig	XIntc_LookupConfig	
XIOModule_ConnectFastHandler	XIntc_ConnectFastHandler	
XIOModule_SetNormalIntrMode	XIntc_SetNormalIntrMode	

Table C-1: **I/O Module Driver API Correspondence *(Cont'd)***

I/O Module Function	Discrete Function	Remark
XIOModule_VoidInterruptHandler	XIntc_VoidInterruptHandler	
XIOModule_InterruptHandler	XIntc_InterruptHandler	
XIOModule_SetOptions	XIntc_SetOptions	
XIOModule_GetOptions	XIntc_GetOptions	
XIOModule_SelfTest	XIntc_SelfTest	
	XIntc_SimulateIntr	Corresponding hardware function not available in I/O Module
XIOModule_Initialize	XUartLite_Initialize	Should only be called once for the entire I/O Module driver
XIOModule_CfgInitialize	XUartLite_CfgInitialize	Should only be called once for the entire I/O Module driver
XIOModule_ResetFifos	XUartLite_ResetFifos	
XIOModule_Send	XUartLite_Send	
XIOModule_Recv	XUartLite_Recv	
XIOModule_IsSending	XUartLite_IsSending	
XIOModule_SetBaudRate		Programmable baud rate not available in discrete hardware
XIOModule_GetStats	XUartLite_GetStats	
XIOModule_ClearStats	XUartLite_ClearStats	
	XUartLite_SelfTest	The I/O Module self-test uses UART TX for output
XIOModule_Uart_EnableInterrupt	XUartLite_EnableInterrupt	
XIOModule_Uart_DisableInterrupt	XUartLite_DisableInterrupt	
XIOModule_SetRecvHandler	XUartLite_SetRecvHandler	
XIOModule_SetSendHandler	XUartLite_SetSendHandler	
XIOModule_Uart_InterruptHandler	XUartLite_InterruptHandler	
XIOModule_Initialize	XTmrCtr_Initialize	Should only be called once for the entire I/O Module driver
XIOModule_Timer_Start	XTmrCtr_Start	Added *Timer* to function name to avoid conflicting names
XIOModule_Timer_Stop	XTmrCtr_Stop	Added *Timer* to function name to avoid conflicting names
XIOModule_GetValue	XTmrCtr_GetValue	
XIOModule_SetResetValue	XTmrCtr_SetResetValue	
XIOModule_GetCaptureValue	XTmrCtr_GetCaptureValue	
XIOModule_IsExpired	XTmrCtr_IsExpired	
XIOModule_Reset	XTmrCtr_Reset	
XIOModule_Timer_SetOptions	XTmrCtr_SetOptions	Added *Timer* to function name to avoid conflicting names

Table C-1: **I/O Module Driver API Correspondence *(Cont'd)***

I/O Module Function	Discrete Function	Remark
XIOModule_Timer_GetOptions	XTmrCtr_GetOptions	Added *Timer* to function name to avoid conflicting names
XIOModule_Timer_GetStats	XTmrCtr_GetStats	Added *Timer* to function name to avoid conflicting names
XIOModule_Timer_ClearStats	XTmrCtr_ClearStats	Added *Timer* to function name to avoid conflicting names
XIOModule_Timer_SelfTest	XTmrCtr_SelfTest	Added *Timer* to function name to avoid conflicting names
XIOModule_SetHandler	XTmrCtr_SetHandler	
XIOModule_Timer_InterruptHandler	XTmrCtr_InterruptHandler	Added *Timer* to function name to avoid conflicting names
XIOModule_Initialize	XGpio_Initialize	Should only be called once for the entire I/O Module driver
XIOModule_CfgInitialize	XGpio_CfgInitialize	Should only be called once for the entire I/O Module driver
	XGpio_SetDataDirection	Separate GPI/GPO, so not necessary
	XGpio_GetDataDirection	Separate GPI/GPO, so not necessary
XIOModule_DiscreteRead	XGpio_DiscreteRead	
XIOModule_DiscreteWrite	XGpio_DiscreteWrite	
XIOModule_DiscreteSet	XGpio_DiscreteSet	
XIOModule_DiscreteClear	XGpio_DiscreteClear	
	XGpio_SelfTest	No self-test of GPI or GPO provided
	XGpio_InterruptGlobalEnable	Corresponding hardware function not available in I/O Module
	XGpio_InterruptGlobalDisable	Corresponding hardware function not available in I/O Module
	XGpio_InterruptEnable	Corresponding hardware function not available in I/O Module
	XGpio_InterruptDisable	Corresponding hardware function not available in I/O Module
	XGpio_InterruptClear	Corresponding hardware function not available in I/O Module
	XGpio_InterruptGetEnabled	Corresponding hardware function not available in I/O Module
	XGpio_InterruptGetStatus	Corresponding hardware function not available in I/O Module

Additional Resources and Legal Notices

Xilinx Resources

For support resources such as Answers, Documentation, Downloads, and Forums, see Xilinx Support.

References

These documents provide supplemental material useful with this user guide:

1. *MicroBlaze Processor Reference Guide* (UG984)
2. *Local Memory Bus (LMB) V10 Product Guide* (PG113)
3. *IP Processor LMB BRAM Interface Controller Product Guide* (PG112)
4. *MicroBlaze Debug Module (MDM) Product Guide* (PG115)
5. *I/O Module Product Guide* (PG111)
6. *7 Series FPGAs Configuration User Guide* (UG470)
7. Xilinx Software Development Kit Help (UG782)
8. *Vivado® Design Suite User Guide: Designing IP Subsystems using IP Integrator* (UG994)
9. *Vivado Design Suite User Guide: Designing with IP* (UG896)
10. *Vivado Design Suite User Guide: Getting Started* (UG910)
11. *Vivado Design Suite User Guide - Logic Simulation* (UG900)
12. *ISE® to Vivado Design Suite Migration Guide* (UG911)
13. *Vivado Design Suite User Guide: Programming and Debugging* (UG908)

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
06/24/2015	2.3	• Moved performance and resource utilization data to the web.
04/01/2015	2.3	• Updated due to core revision.
04/02/2014	2.2	• Updated due to core revision. • Option to enable Debug UART added. • Additional memory sizes added.
12/18/2013	2.1	• Updated due to core revision. • Synchronization flip-flops added on asynchronous interrupt inputs.
10/02/2013	2.0	• Document version number advanced to match the core version number. • Updated due to core revision. • Added description of board interfaces.
03/20/2013	1.0	• Initial release as a Product Guide; replaces PG048. No other documentation changes.

Please Read: Important Legal Notices