# XPS Mutex (v1.00c)



## Introduction

In a multi processor environment, the processors share common resources. The mutex provides a mechanism for mutual exclusion to enable one process to gain exclusive access to a particular resource.

XPS Mutex core contains a configurable number of mutexes. Each of these can be associated with a 32-bit User configuration register to store arbitrary data.

## Features

- PLB interface is based on PLB v4.6 specification
- Configurable number of PLB interfaces from 1 to 8
- Configurable asynchronous or synchronous interface operation
- Configurable USER register
- Configurable number of mutexes
- Configurable CPUID width
- Configurable enhanced security through hardware identification support

| LogiCORE™ Facts | | |
|---|---|---|
| **Core Specifics** | | |
| Supported Device Family | Spartan®-3, Spartan-3E, Spartan-6, Spartan-3A/3A DSP/3AN, Automotive Spartan-3/3A/3A DSP/ 3E, Virtex®-4, Virtex-4Q, Virtex-4QV, Virtex-5, Virtex-6 | |
| Version of core | xps_mutex | v1.00c |
| Resources Used[1] | | |
| | Min | Max |
| Slices | 110 | 560 |
| LUTs | 115 | 460 |
| FFs | 125 | 730 |
| Block RAMs | 0 | 0 |
| Special Features | None | |
| **Provided with Core** | | |
| Documentation | Product Specification | |
| Design File Formats | VHDL | |
| Constraints File | N/A | |
| Verification | N/A | |
| Instantiation Template | N/A | |
| **Design Tool Requirements** | | |
| Xilinx Implementation Tools | ISE® 11.1 or later | |
| Verification | N/A | |
| Simulation | ModelSim PE/SE 6.4b or later | |
| Synthesis | XST | |
| **Support** | | |
| Provided by Xilinx, Inc. | | |

1. Resources for Virtex-4 implementation with 16 mutexes and 8 bits CPUID. Minimum has one and maximum eight PLB interface

## Functional Description

The XPS Mutex core contains a configurable number of mutexes. Each of these can be associated with a 32-bit user configuration register to store arbitrary data.

In a multi processor environment, the processors share common resources. The mutex provides a mechanism for mutual exclusion to enable one process to gain exclusive access to a particular resource.

## XPS Mutex I/O Signals

The XPS Mutex has a configurable number of PLB interfaces that are used to connect it to the rest of the system. All PLB interfaces can be independently configured and contain the signals listed in Table 1 where <x> denotes the interface number (0 to 7).

*Table 1:* **XPS Mutex PLBv46 I/O Signal Description**

| Port | Signal Name | Interface | I/O | Initial State | Description |
|---|---|---|---|---|---|
| **System Signals** | | | | | |
| P1 | SPLB<x>_Clk | System | I | - | PLB clock |
| P2 | SPLB<x>_Rst | System | I | - | PLB reset, active high |
| **PLB Interface Signals** | | | | | |
| P3 | PLB<x>_ABus[0:31] | PLB | I | - | PLB address bus |
| P4 | PLB<x>_PAValid | PLB | I | - | PLB primary address valid |
| P5 | PLB<x>_masterID[0: C_SPLB_MID_WIDTH - 1] | PLB | I | - | PLB current master identifier |
| P6 | PLB<x>_RNW | PLB | I | - | PLB read not write |
| P7 | PLB<x>_BE[0: (C_SPLB_DWIDTH/8) - 1] | PLB | I | - | PLB byte enables |
| P8 | PLB<x>_size[0:3] | PLB | I | - | PLB size of requested transfer |
| P9 | PLB<x>_type[0:2] | PLB | I | - | PLB transfer type |
| P10 | PLB<x>_wrDBus[0: C_SPLB_DWIDTH - 1] | PLB | I | - | PLB write data bus |
| **Unused PLB Interface Signals** | | | | | |
| P11 | PLB<x>_UABus[0:31] | PLB | I | - | PLB upper address bits |
| P12 | PLB<x>_SAValid | PLB | I | - | PLB secondary address valid |
| P13 | PLB<x>_rdPrim | PLB | I | - | PLB secondary to primary read request indicator |
| P14 | PLB<x>_wrPrim | PLB | I | - | PLB secondary to primary write request indicator |
| P15 | PLB<x>_abort | PLB | I | - | PLB abort bus request |
| P16 | PLB<x>_busLock | PLB | I | - | PLB bus lock |
| P17 | PLB<x>_MSize[0:1] | PLB | I | - | PLB data bus width indicator |
| P18 | PLB<x>_lockErr | PLB | I | - | PLB lock error |
| P19 | PLB<x>_wrBurst | PLB | I | - | PLB burst write transfer |

*Table 1:* **XPS Mutex PLBv46 I/O Signal Description *(Contd)***

| Port | Signal Name | Interface | I/O | Initial State | Description |
|---|---|---|---|---|---|
| P20 | PLB<x>_rdBurst | PLB | I | - | PLB burst read transfer |
| P21 | PLB<x>_wrPendReq | PLB | I | - | PLB pending bus write request |
| P22 | PLB<x>_rdPendReq | PLB | I | - | PLB pending bus read request |
| P23 | PLB<x>_wrPendPri[0:1] | PLB | I | - | PLB pending write request priority |
| P24 | PLB<x>_rdPendPri[0:1] | PLB | I | - | PLB pending read request priority |
| P25 | PLB<x>_reqPri[0:1] | PLB | I | - | PLB current request priority |
| P26 | PLB<x>_TAttribute[0:15] | PLB | I | - | PLB transfer attribute |
| **PLB Slave Interface Signals** | | | | | |
| P27 | Sl<x>_addrAck | PLB | O | 0 | Slave address acknowledge |
| P28 | Sl<x>_SSize[0:1] | PLB | O | 0 | Slave data bus size |
| P29 | Sl<x>_wait | PLB | O | 0 | Slave wait |
| P30 | Sl<x>_rearbitrate | PLB | O | 0 | Slave bus rearbitrate |
| P31 | Sl<x>_wrDAck | PLB | O | 0 | Slave write data acknowledge |
| P32 | Sl<x>_wrComp | PLB | O | 0 | Slave write transfer complete |
| P33 | Sl<x>_rdDBus[0: C_SPLB_DWIDTH - 1] | PLB | O | 0 | Slave read data bus |
| P34 | Sl<x>_rdDAck | PLB | O | 0 | Slave read data acknowledge |
| P35 | Sl<x>_rdComp | PLB | O | 0 | Slave read transfer complete |
| P36 | Sl<x>_MBusy[0: C_SPLB_NUM_MASTERS - 1] | PLB | O | 0 | Slave busy |
| P37 | Sl<x>_MWrErr[0: C_SPLB_NUM_MASTERS - 1] | PLB | O | 0 | Slave write error |
| P38 | Sl<x>_MRdErr[0: C_SPLB_NUM_MASTERS - 1] | PLB | O | 0 | Slave read error |
| **Unused PLB Slave Interface Signals** | | | | | |
| P39 | Sl<x>_wrBTerm | PLB | O | 0 | Slave terminate write burst transfer |
| P40 | Sl<x>_rdWdAddr[0:3] | PLB | O | 0 | Slave read word address |
| P41 | Sl<x>_rdBTerm | PLB | O | 0 | Slave terminate read burst transfer |
| P42 | Sl<x>_MIRQ[0: C_SPLB_NUM_MASTERS - 1] | PLB | O | 0 | Master interrupt request |

## XPS Mutex Design Parameters

The XPS Mutex design is parameterized to tailor it for different systems. This allows the user to configure a design that utilizes the resources required by the system only and that operates with the best possible performance. The features that can be parameterized in the XPS Mutex design are shown in Table 2. The PLB related generics, G2 through G9, are separately configured for each interface.

*Table 2:* **XPS Mutex Design Parameters**

| Generic | Feature/Description | Parameter Name | Allowable Values | Default Value | VHDL Type |
|---|---|---|---|---|---|
| **System Parameter** | | | | | |
| G1 | Target FPGA family | C_FAMILY | spartan3, aspartan3, spartan3e, aspartan3e, spartan3a, aspartan3a, spartan3adsp, aspartan3adsp, spartan6, virtex4, qrvirtex4, qvirtex4, virtex5, virtex6 | virtex-5 | string |
| **PLB Parameters** | | | | | |
| G2 | PLB Base Address | C_SPLB<x>_BASE ADDR | Valid Address[1] | None[3] | std_logic_ vector |
| G3 | PLB High Address | C_SPLB<x>_HIGHA DDR | Valid Address[2] | None[3] | std_logic_ vector |
| G4 | PLB least significant address bus width | C_SPLB<x>_AWIDT H | 32 | 32 | integer |
| G5 | PLB data width | C_SPLB<x>_DWIDT H | 32, 64, 128 | 32 | integer |
| G6 | Selects point-to-point or shared bus topology | C_SPLB<x>_P2P | 0 = Shared Bus Topology<br>1 = Point-to-Point Bus Topology[4] | 0 | integer |
| G7 | PLB Master ID Bus Width | C_SPLB<x>_MID_ WIDTH | $\log_2$(C_SPLB_NUM_ MASTERS) with a minimum value of 1 | 1 | integer |
| G8 | Number of PLB Masters | C_SPLB<x>_NUM_ MASTERS | 1 - 16 | 1 | integer |
| G9 | Support Bursts | C_SPLB<x>_SUPP ORT_BURSTS | 0 | 0 | integer |
| G10 | Width of the Slave Data Bus | C_SPLB_NATIVE_ DWIDTH | 32 | 32 | integer |
| **XPS Mutex Parameters** | | | | | |
| G11 | Specify if PLB interfaces are synchronous or asynchronous | C_ASYNC_CLKS | 0 - 1 | 0 | Integer |

*Table 2:* **XPS Mutex Design Parameters *(Contd)***

| Generic | Feature/Description | Parameter Name | Allowable Values | Default Value | VHDL Type |
|---|---|---|---|---|---|
| G13 | Number of PLB interfaces | C_NUM_PLB | 1 - 8 | 1 | Integer |
| G14 | If the 32-bit USER register associated with a mutex should be available | C_ENABLE_USER | 0 - 1 | 32 | Integer |
| G15 | Number of bits used for the CPUID field | C_OWNER_ID_WIDTH | 8 | 8 | Integer |
| G16 | If hardware protection of a mutex should be enabled besides the CPUID (if available) | C_ENABLE_HW_PROT | 0 - 1 | 0 | Integer |
| G17 | Number of mutexes that shall be contained inside the core | C_NUM_MUTEX | 1 - | 16 | Integer |

**Notes:**
1. The user must set the values. The C_BASEADDR must be a multiple of the range, where the range is C_HIGHADDR - C_BASEADDR + 1.
2. C_HIGHADDR - C_BASEADDR must be a power of 2 greater than equal to C_BASEADDR + 0xF.
3. No default value will be specified to insure that the actual value is set, i.e., if the value is not set, a compiler error will be generated.
4. Value of '1' is not supported in this core.

### Allowable Parameter Combinations

The address range specified by C_BASEADDR and C_HIGHADDR must be a power of 2, and must be at least 0xF.

For example, if C_BASEADDR = 0xE0000000, C_HIGHADDR must be at least = 0xE000000F.

## XPS Mutex Parameter - Port Dependencies

The dependencies between the XPS Mutex core design parameters and I/O signals are described in Table 3. In addition, when certain features is deselected, the related logic will no longer be a part of the design. The unused input and output signals are set to a specified value.

.

*Table 3:* **XPS Mutex Parameter-Port Dependencies**

| Generic or Port | Name | Affects | Depends | Relationship Description |
|---|---|---|---|---|
| Design Parameters | | | | |
| G5 | C_SPLB<x>_DWIDTH | P7, P10, P33 | - | Affects the number of bits in data bus |
| G7 | C_SPLB<x>_MID_WIDTH | P5 | G8 | This value is calculated as: $\log_2$(C_SPLB<x>_NUM_MASTERS) with a minimum value of 1 |
| G8 | C_SPLB<x>_NUM_MASTERS | P36, P37, P38, P42 | - | Affects the number of PLB masters |

*Table 3:* **XPS Mutex Parameter-Port Dependencies *(Contd)***

| Generic or Port | Name | Affects | Depends | Relationship Description |
|---|---|---|---|---|
| I/O Signals | | | | |
| P5 | PLB<x>_masterID[0: C_SPLB_MID_WIDTH - 1] | - | G7 | Width of the PLB<x>_masterID varies according to C_SPLB<x>_MID_WIDTH |
| P7 | PLB<x>_BE[0: (C_SPLB_DWIDTH/8) -1] | - | G5 | Width of the PLB<x>_BE varies according to C_SPLB<x>_DWIDTH |
| P10 | PLB<x>_wrDBus[0: C_SPLB_DWIDTH - 1] | - | G5 | Width of the PLB<x>_wrDBus varies according to C_SPLB<x>_DWIDTH |
| P33 | SI<x>_rdDBus[0: C_SPLB_DWIDTH - 1] | - | G5 | Width of the SI<x>_rdDBus varies according to C_SPLB<x>_DWIDTH |
| P36 | SI<x>_MBusy[0: C_SPLB_NUM_MASTERS - 1] | - | G8 | Width of the SI<x>_MBusy varies according to C_SPLB<x>_NUM_MASTERS |
| P37 | SI<x>_MWrErr[0: C_SPLB_NUM_MASTERS - 1] | - | G8 | Width of the SI<x>_MWrErr varies according to C_SPLB<x>_NUM_MASTERS |
| P38 | SI<x>_MRdErr[0: C_SPLB_NUM_MASTERS - 1] | - | G8 | Width of the SI<x>_MRdErr varies according to C_SPLB<x>_NUM_MASTERS |
| P42 | SI<x>_MIRQ[0: C_SPLB_NUM_MASTERS - 1] | - | G8 | Width of the SI<x>_MIRQ varies according to C_SPLB<x>_NUM_MASTERS |

## XPS Mutex Register Descriptions

Each interface of the XPS Mutex core can access all mutexes. Only one interface at the time can access any of the mutexes, i.e. all other PLB interfaces are blocked while one is accessing any of the mutexes. PLB interface arbitration has fixed priority where PLB0 has the highest priority and PLB7 the lowest.

Table 4 shows all the XPS Mutex registers and their addresses offsets.

*Table 4:* **XPS Mutex Registers**

| Base Address + Offset (hex) | Register Name | Access Type | Default Value (hex) | Description |
|---|---|---|---|---|
| C_BASEADDR + 0x0 | MUTEX | R/W | 0 | Mutex register for mutex ownership |
| C_BASEADDR + 0x4 | USER | N/A | 0 | USER configuration register. |
| C_BASEADDR + 0x8 to 0xFC | Reserved | | | Reserved for future use |

### XPS Mutex Register (MUTEX)

The MUTEX register contains one mandatory and two optional bit fields. The LOCK bit is required since this bit determines if the mutex is in locked or released state. CPUID is usually included to control access of who may manipulate the mutex. It is only the owner of the mutex that may release it. For extra safety an optional HWID field is also available. The HWID bits are not accessible by the user and are handled implicitly in the background. HWID contains the PLB master ID and on which port the PLB

master is attached. This guarantees that no other processor can fake the CPUID and gain access over the mutex. Bit assignment in the MUTEX register is described in Table 6.

CPUID is a unique identification value assigned by the tools to software that executes on each processor. Since CPUID is only assigned to software created from within EDK, any other master that accesses the mutex must be manually assigned a unique number that does not interfere with the others. Examples of this are external processors and hardware IPs other than MicroBlaze™ and PowerPC® processors. Each processor has its allocated CPUID listed in xparameters.h.

### Mutex lock and release process

The steps needed to lock and release a mutex (for a free mutex the MUTEX register is zero):

- Write <CPUID & 1> to the MUTEX register. If the mutex is free the lock bit will be set to one and the CPUID field will be update with the new CPUID. If C_ENABLE_HW_PROT is enabled, the PLB_masterID and interface number is also stored for enhanced protection. Should the mutex already be locked the access is ignored.
- Read back the MUTEX register to verify that the mutex has been locked by the current CPU by comparing the value with the written CPUID, if not retry step 1 until ownership has been granted.
- Manipulate the shared resource that is protected by the mutex.
- Release mutex by writing <CPUID & 0> to the mutex register. If C_ENABLE_HW_PROT is enabled, the PLB_masterID and interface number are also taken into account. The mutex will automatically set the MUTEX register to zero.

If the "wrong" processor tries to free the mutex with C_ENABLE_HW_PROT active with the correct CPUID the operation will be ignored since both the HWID and CPUID must match for the operation to be successful. Also, the operation is ignored if the "right" processor writes the wrong CPUID.

*Table 5:* **XPS Mutex Write Data Register**

| Reserved | | CPUID | | Lock |
|---|---|---|---|---|
| 0 | 22 | 23 | 30 | 31 |

*Table 6:* **XPS Mutex Write Data Register Bit Definitions**

| Bit(s) | Name | Core Access | Reset Value | Description |
|---|---|---|---|---|
| 0 - 22 | Reserved | N/A | 0 | Reserved for future use. |
| 23 - 30 | CPUID | R/W | - | Unique processor ID number. |
| 31 | LOCK | R/W | - | Lock status: 0 = free, 1 = Mutex currently owned by CPUID. |

### XPS Mutex User Configuration Register (USER)

The USER configuration is used to store a 32-bit value associated with a mutex. It can contain any arbitrary information. Bit assignment in the USER register is described in Table 8.

*Table 7:* **XPS Mutex User Configuration Register (USER)**

| USER | |
|---|---|
| 0 | 31 |

*Table 8:* **XPS Mutex Read Data Register Bit Definitions**

| Bit(s) | Name | Core Access | Reset Value | Description |
|---|---|---|---|---|
| 0 - 31 | USER | R/W | - | User configuration register |

## Design Implementation

### Target Technology

The intended target technology is an FPGA in one of the following families: Spartan-3, Spartan-3E, Spartan-3A, Spartan-3A DSP, Automotive Spartan-3/3A/3E/3A DSP, Virtex-4, Virtex-4QV, Virtex-4Q, Virtex-5, Virtex-6.

## Reference Documents

1. IBM CoreConnect128-Bit Processor Local Bus, Architectural Specification (v4.6).

## Support

Xilinx provides technical support for this LogiCORE 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*.

## Revision History

| Date | Version | Revision |
|---|---|---|
| 07/04/07 | 1.0 | Initial Xilinx release. |
| 02/20/08 | 1.1 | Updated CPUID information. |
| 7/25/08 | 1.2 | Added QPro® Virtex-4 Hi-Rel and QPro Virtex-4 Rad Tolerant support. |
| 10/02/08 | 1.3 | Initial version for v1.00b |
| 4/24/09 | 1.4 | Replaced references to supported device families and tool name(s) with hyperlink to PDF file. |
| 6/26/09 | 1.5 | Updated for EDK_L 11.2; created v1.00c. |

## Notice of Disclaimer