XPS PS2 Controller (v1.01a)

DS707 April 24, 2009 **Product Specification**

Introduction

The XPS PS2 Controller is a PLB (Processor Local Bus) slave that is designed to control PS2 devices such as keyboard and mouse. The PS2 protocol is a simple bidirectional serial protocol.

Features

- Connects as a 32-bit slave on PLB V4.6 bus of 32, 64 or 128 bit data width
- Configurable as single or dual port PS2 controller
- Supports two PS2 devices, each controlled by separate set of eight byte-wide registers
- Two separate interrupts for each of the ports

LogiCORE™ Facts		
Core Specifics		
Supported Device Family	See EDK Supported Device Families.	
Version of Core	xps_ps2	v1.00a
Resources Used		
	Min	Max
SLICES	Refer to the Table 14, Table 15, Table 16, Table 17 and Table 18	
LUTs		
FFs		
Block RAMs	N/A	
Special Features	N/A	
Provided with Core		
Documentation	Product Specification	
Design File Formats	VHDL	
Constraints File	N/A	
Verification	N/A	
Instantiation Template	N/A	
Reference Designs & Application notes	N/A	
Design Tool Requirements		
Xilinx Implementation Tools	See Tools for requirements.	
Verification		
Simulation		
Synthesis		
Support		
Provided by Xilinx, Inc.		

Functional Description

The XPS PS2 Controller is a slave IP core designed to control two PS2 devices such as a keyboard and a mouse. To control the two PS2 devices, it uses simple state machines and shift registers. Each of the PS2 ports is controlled by a separate set of four byte-wide registers. For transmitting data, a byte is written to the transmit register and then the Serial Interface Engine serializes the data and transmits to the PS2 device. Transmit status registers and interrupts indicate whether the transmission is complete and if there are any errors reported. While receiving data, the Serial Interface Engine receives serial data from the PS2 device and writes into the receive register. Similar to the transmit status registers, receive status registers and interrupts indicate whether data has been received from the PS2 device. Any errors in the received data are also reported. The XPS PS2 Controller generates interrupts upon various transmit and receive conditions.The XPS PS2 Controller can be operated in a polled mode or an interrupt driven mode.

PS2 Communication

The PS2 protocol consists of host-to-device and device-to-host communication. In the description below "host" implies the XPS PS2 Controller and "device" implies any PS2 device, which would be a keyboard or a mouse.

The PS2 protocol is a bidirectional synchronous serial protocol. The data and the clock are the two signals through which communication between the device and the host happens. The host is given the ultimate control of the data bus. The basic states which can be defined based on the status of the data and clock lines are:

- Idle State - Data is high and clock is high. This is the only state where the PS2 device is allowed to start transmission of data (during device-to-host communication).
- Communication Inhibited State - Data high and clock low. The device always generates the clock signal. But since the host has the ultimate control of the bus and may inhibit communication anytime, it must pull the clock signal low and inhibit the transmission by the device and then initiate transmission from its side.
- Host Request-to-Send State - Data is low and clock is high. The host after inhibiting the communication, will pull the data line low and release the clock inline, signalling the device that host would transmit data.

All the data is transmitted one byte at a time and each byte is sent in a frame consisting of 11-12 bits (depending on whether it is host-to-device or device-to-host communication). These bits are:

- 1 start bit. This is always 0
- 8 data bits, least significant bit first
- 1 parity bit (odd parity)
- 1 stop bit. This is always 1
- 1 acknowledge bit (host-to-device communication only)

The data sent from the device to host is read on the falling edge and the data sent from the host to device is read on the rising edge. The clock frequency must be in the range 10-16.7 kHz. This means clock must be high for 30 - 50 microseconds and low for 30 - 50 microseconds. The keyboard, mouse or host emulator should modify/sample the data line in the middle of each cell, i.e. 15 - 25 microseconds after the appropriate clock transition.

Device-to-Host Communication

The device-to-host communication happens over 11-bit frames. When the keyboard or mouse wants to send information, it first checks the clock line to make sure it's at a high logic level. If it's not, the host is inhibiting communication and the device must buffer any to-be-sent data until the host releases clock. The clock line must be continuously high for at least 50 microseconds before the device can begin to transmit data.

The host may inhibit communication at any time by pulling the clock line low for at least 100 microseconds. If a transmission is inhibited before the 11th clock pulse, the device must abort the current transmission and prepare to retransmit the current "chunk" of data when host releases clock. A "chunk" of data could be a make code, break code, device ID, mouse movement packet, etc. For example, if a keyboard is interrupted while sending the second byte of a two-byte break code, it will need to retransmit both bytes of that break code, not just the one that was interrupted.

If the host pulls clock low before the first high-to-low clock transition, or after the falling edge of the last clock pulse, the PS2 device does not need to retransmit any data. However, if new data is created that needs to be transmitted, it will have to be buffered until the host releases clock.

Figure 1 illustrates the above mentioned device-to-host communication

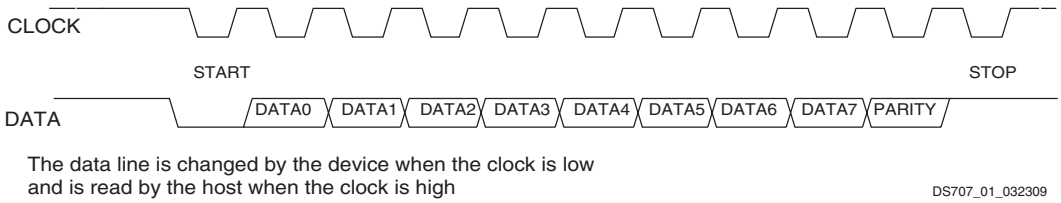

Figure 1: **Device-to-Host Communication**

Host-to-Device Communication

The host-to-device communication happens over 12-bit frames. Since the PS2 device always generates the clock signal, the host whenever it wants to send data, must inhibit communication by pulling the clock low for at least 100 microseconds. It must then apply a "request-to-send" by pulling data low while releasing the clock signal.

The device should check for this state at intervals not to exceed 10 milliseconds. When the device detects this state, it will begin generating clock signals and clock in eight data bits and one stop bit. The host changes the data line only when the clock line is low, and data is read by the device when clock is high. After the stop bit is received, the device will acknowledge the received byte by bringing the data line low and generating one last clock pulse. If the host does not release the data line after the 11th clock pulse, the device will continue to generate clock pulses until the data line is released (the device will then generate an error.)

The host may abort transmission at time before the 11th clock pulse (acknowledge bit) by holding clock low for at least 100 microseconds.

There are two timings that are needed to be taken care by the state machines. The time it the device to begin generating clock pulses after the host initially takes the Clock line low, which must be no greater than 15 milliseconds. And the time it takes for the packet to be sent, which must be no greater than 2 milliseconds. If either of the above time limits is not met, the host should generate an error. If the command sent by the host requires a response, that response must be received no later than 20 milliseconds after the host releases the Clock line. If this does not happen, the host should generate an error.

Figure 2 illustrates the above mentioned host-to-device communication

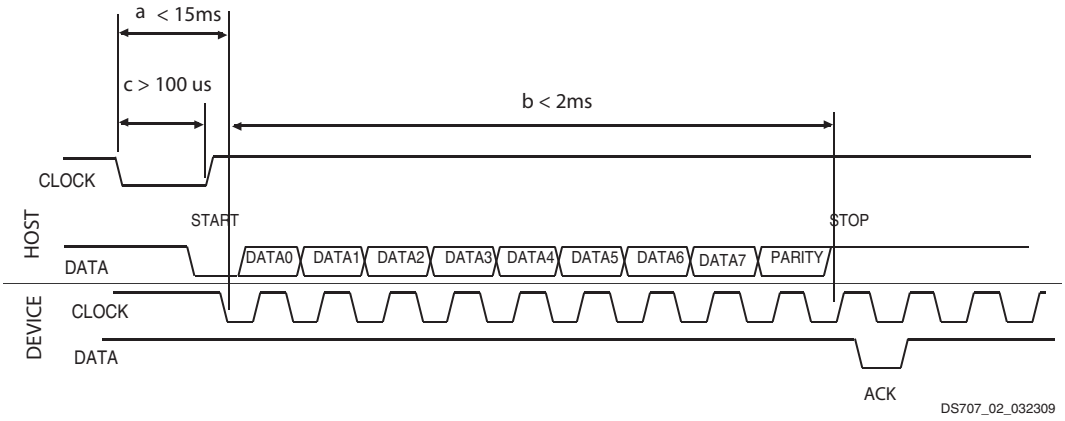

Figure 2: **Host-to-device communication**

The major interface and modules, for each of the ports, of the XPS PS2 Controller are shown in Figure 3 and described in the subsequent sections. These modules are:

- PLB Interface Module
- Interrupt Service Controller
- Serial Interface Engine (SIE)
- Internal Registers

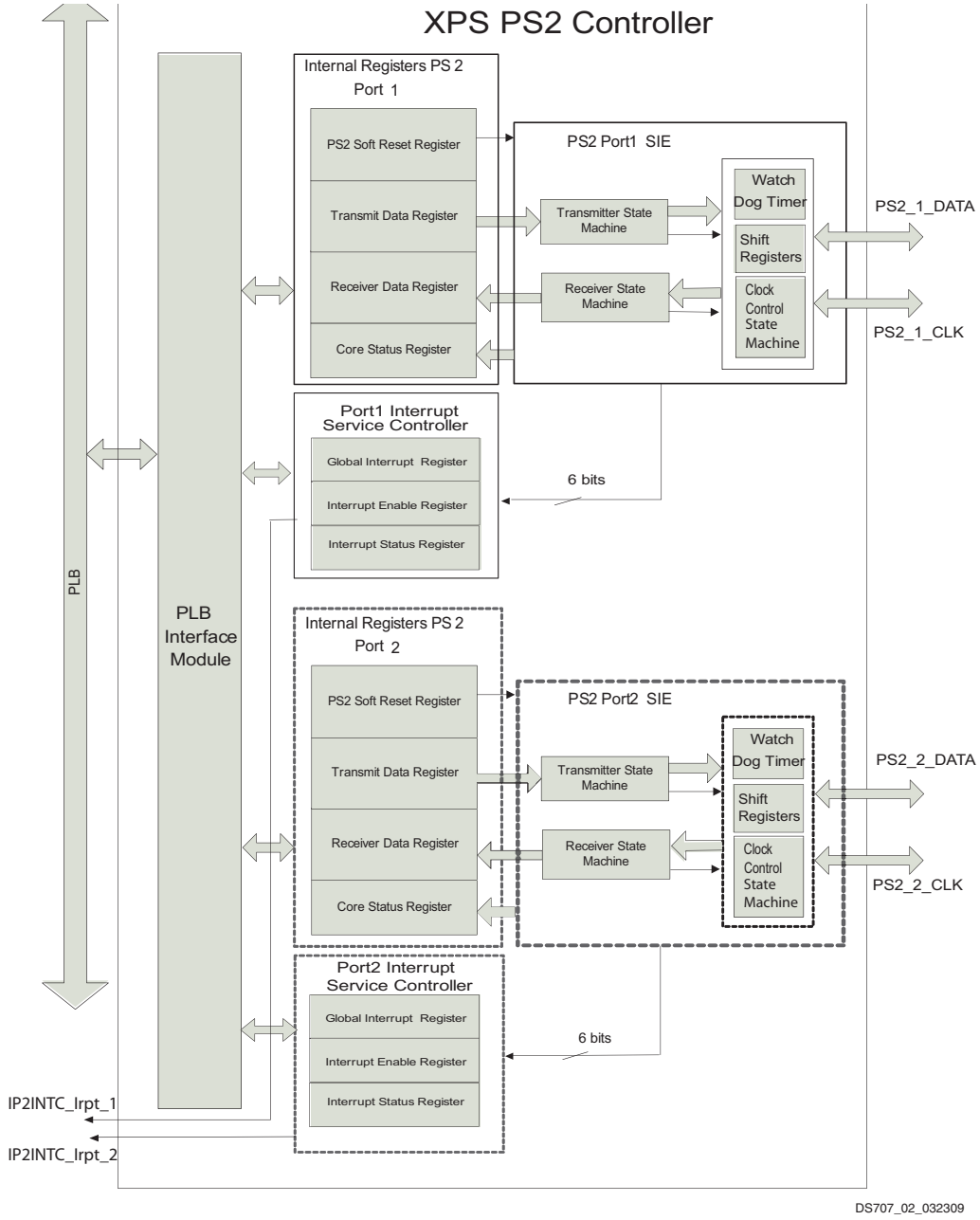

Figure 3: **XPS PS2 Controller Block Diagram**

PLB Interface Module

The PLB Interface Module is a bi-directional interface between XPS PS2 Controller IP core and the PLB. To simplify the process of attaching a XPS PS2 Controller to the PLB, the core makes use of a portable, pre-designed bus interface called PLB Interface Module, that takes care of the bus interface signals, bus protocols and other interface issues. The base element of the PLB Interface Module is slave attachment, which provides the basic functionality of PLB slave operation.

Interrupt Service Controller

The Interrupt Service Controller is a continuation of the Xilinx family of IBM CoreConnect compatible LogiCORE products. It provides interrupt capture support for the connected IP function. The interrupts from XPS PS2

Controller are connected to the Interrupt Service Controller and the corresponding bits in the Interrupt Status Register get updated. Interrupt Service Controller provides following functions:

- Parameterized number of interrupts needed by the IP.
- Provides both Interrupt Status Register (ISR) and Interrupt Enable Register (IER) functions for the user IP. Depending on which interrupt bits are enabled in the IER, the corresponding interrupts from the IP would be or-ed and a single interrupt would be generated.

Serial Interface Engine

The Serial Interface Engine has the following modules:

- Transmit State Machine - There are 2 separate transmit state machines for each of the PS2 ports. This helps each of the ports to transmit simultaneously irrespective of the other port. The transmit state machine serializes the data written into the transmit data register and sends it over the data line as per the PS2 protocol as mentioned above in the Host-to-Device Communication section.
- Receive State Machine - There are 2 separate receive state machines for each of the PS2 ports. This helps each of the ports to receive simultaneously irrespective of the other port. The receive state machine does the serial-to-parallel conversion of the serial data received on the data line and writes into the receive data register and sends it over the data line as per the PS2 protocol as mentioned above in the Device-to-Host Communication section.
- Shift Registers - The parallel-to-serial and serial-to-parallel conversion of data by the transmit state machines and receive state machine respectively, is done by using these shift registers.
- Clock Control State Machine - This state machine detects the rise and fall of the clock line. This edge detection is required for the transmit and receive state machines to perform the operation. The PS2 protocol is greatly dependent on the edge of the clock.
- Watch Dog Timer- The watch dog timer keeps a watch on the clock line while transmitting. If the clock line goes into the pull-up mode in between transmission, the watch dog timer sets the corresponding interrupt. If there are no transitions on the clock for 200 microseconds, the watch timer would set the interrupt.

Internal Registers

The XPS PS2 controller has eight internal registers, four registers for each of the PS2 ports. The whole operation happens through the internal registers. For transmitting data to the PS2 device, the data needs to be written into the transmit register and similarly the receive register would get updated as soon as the data is received from the PS2 device. A more detailed description of these registers is given in the XPS PS2 Controller Register Descriptions section.

XPS PS2 Controller I/O Signals

The XPS PS2 Controller I/O signals are listed and described in Table 1. All signals are active high.

Table 1: **XPS PS2 Controller I/O Signal Description**

Port	Signal Name	Interface	I/O	Initial State	Description
System Signals					
P1	SPLB_Clk	System	I	-	PLB clock
P2	SPLB_Rst	System	I	-	PLB reset
PLB Slave Interface Input Signals					
P3	PLB_ABus[0: C_SPLB_AWIDTH - 1]	PLB	I	-	PLB address bus
P4	PLB_PAValid	PLB	I	-	PLB primary address valid
P5	PLB_masterID[0: C_SPLB_MID_WIDTH - 1]	PLB	I	-	PLB current master identifier
P6	PLB_RNW	PLB	I	-	PLB read not write
P7	PLB_BE[0: (C_SPLB_DWIDTH/8) - 1]	PLB	I	-	PLB byte enables
P8	PLB_size[0:3]	PLB	I	-	PLB size of requested transfer
P9	PLB_type[0:2]	PLB	I	-	PLB transfer type
P10	PLB_wrDBus[0: C_SPLB_DWIDTH - 1]	PLB	I	-	PLB write data bus
Unused PLB Slave Interface Input Signals					
P11	PLB_UABus[0: 31]	PLB	I	-	PLB upper address bits
P12	PLB_SAValid	PLB	I	-	PLB secondary address valid
P13	PLB_rdPrim	PLB	I	-	PLB secondary to primary read request indicator
P14	PLB_wrPrim	PLB	I	-	PLB secondary to primary write request indicator
P15	PLB_abort	PLB	I	-	PLB abort bus request
P16	PLB_busLock	PLB	I	-	PLB bus lock
P17	PLB_MSize	PLB	I	-	PLB data bus width indicator
P18	PLB_lockErr	PLB	I	-	PLB lock error
P19	PLB_wrBurst	PLB	I	-	PLB burst write transfer
P20	PLB_rdBurst	PLB	I	-	PLB burst read transfer
P21	PLB_wrPendReq	PLB	I	-	PLB pending bus write request
P22	PLB_rdPendReq	PLB	I	-	PLB pending bus read request
P23	PLB_wrPendPri[0:1]	PLB	I	-	PLB pending write request priority

Table 1: **XPS PS2 Controller I/O Signal Description *(Cont.)***

Port	Signal Name	Interface	I/O	Initial State	Description
P24	PLB_rdPendPri[0:1]	PLB	I	-	PLB pending read request priority
P25	PLB_reqPri[0:1]	PLB	I	-	PLB current request priority
P26	PLB_TAttribute[0:15]	PLB	I	-	PLB transfer attribute
PLB Slave Interface Output Signals					
P27	Sl_addrAck	PLB	O	0	Slave address acknowledge
P28	Sl_SSize[0:1]	PLB	O	0	Slave data bus size
P29	Sl_wait	PLB	O	0	Slave wait
P30	Sl_rearbitrate	PLB	O	0	Slave bus rearbitrate
P31	Sl_wrDAck	PLB	O	0	Slave write data acknowledge
P32	Sl_wrComp	PLB	O	0	Slave write transfer complete
P33	Sl_rdDBus[0: C_SPLB_DWIDTH - 1]	PLB	O	0	Slave read data bus
P34	Sl_rdDAck	PLB	O	0	Slave read data acknowledge
P35	Sl_rdComp	PLB	O	0	Slave read transfer complete
P36	Sl_MBusy[0: C_SPLB_NUM_MASTERS - 1]	PLB	O	0	Slave busy
P37	Sl_MWrErr[0: C_SPLB_NUM_MASTERS - 1]	PLB	O	0	Slave write error
P38	Sl_MRdErr[0: C_SPLB_NUM_MASTERS - 1]	PLB	O	0	Slave read error
Unused PLB Slave Interface Output Signals					
P39	Sl_wrBTerm	PLB	O	0	Slave terminate write burst transfer
P40	Sl_rdWdAddr[0:3]	PLB	O	0	Slave read word address
P41	Sl_rdBTerm	PLB	O	0	Slave terminate read burst transfer
P42	Sl_MIRQ[0: C_SPLB_NUM_MASTERS - 1]	PLB	O	0	Master interrupt request
XPS PS2 Controller Signals					
P43	PS2_1_DATA	PS2	I/O	Z	Bi-directional Port1 Data
P44	PS2_1_CLK	PS2	I/O	Z	Bi-directional Port1 Clock
P45	PS2_2_DATA	PS2	I/O	Z	Bi-directional Port2 Data

Table 1: **XPS PS2 Controller I/O Signal Description *(Cont.)***

Port	Signal Name	Interface	I/O	Initial State	Description
P46	PS2_2_CLK	PS2	I/O	Z	Bi-directional Port2 Clock
P47	IP2INTC_Irpt_1	PS2	O	0	Active High Level Triggered Interrupt Signal from Port1
P48	IP2INTC_Irpt_2	PS2	O	0	Active High Level Triggered Interrupt Signal from Port2

XPS PS2 Controller Design Parameters

To allow the user to create a XPS PS2 Controller that is uniquely tailored for the user's system, certain features are parameterizable in the XPS PS2 Controller design. This allows the user to have a design that utilizes only the resources required by the system and runs at the best possible performance. The features that are parameterizable in the XPS PS2 Controller core are as shown in Table 2.

Table 2: **XPS PS2 Controller Design Parameters**

Generic	Feature/Description	Parameter Name	Allowable Values	Default Value	VHDL Type
System Parameter					
G1	Target FPGA family	C_FAMILY	See C_FAMILY parameter values.		string
PLB Parameters					
G2	XPS PS2 Controller Base Address	C_BASEADDR	Valid Address(2)	None(1)	std_logic_vector
G3	XPS PS2 Controller High Address	C_HIGHADDR	Valid Address(2)	None(1)	std_logic_vector
G4	PLB address width	C_SPLB_AWIDTH	32	32	integer
G5	PLB data width	C_SPLB_DWIDTH	32, 64, 128	32	integer
G6	Selects point-to-point or shared PLB topology	C_SPLB_P2P	0 = Shared Bus Topology	0	integer
G7	PLB Master ID Bus Width	C_SPLB_MID_WIDTH	$\log_2$(C_SPLB_NUM_MASTERS) with a minimum value of 1	1	integer
G8	Number of PLB Masters	C_SPLB_NUM_MASTERS	1 - 16	1	integer
G9	Width of the Slave Data Bus	C_SPLB_NATIVE_DWIDTH	32	32	integer
G10	Enable burst support	C_SPLB_SUPPORT_BURSTS	0	0	integers
XPS PS2 Controller Features					

Table 2: **XPS PS2 Controller Design Parameters *(Cont.)***

Generic	Feature/Description	Parameter Name	Allowable Values	Default Value	VHDL Type
G11	Use dual channel	C_IS_DUAL	0/1	0	integer
G12	PLB Clock Frequency	C_SPLB_CLK_FREQ_HZ	Valid Frequency(3)	100000000	integer

Notes:
1. 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
2. C_BASEADDR must be a multiple of the range size, where the range size is C_HIGHADDR - C_BASEADDR + 1 and must be a power of two large enough to accommodate all of the registers
3. Valid Frequency in Hertz, depending on the device family. For spartan3 devices, the frequency can be a maximum of 100MHz. For virtex4 devices, a maximum of 125MHz and for virtex5 devices, a maximum of 150MHz

Allowable Parameter Combinations

The address-range size specified by C_BASEADDR and C_HIGHADDR must be a power of 2, and must be at least 0x40 in the single port mode. The address range should be at least 0x1040 in the dual port mode, so that the registers associated with the second port are also accommodated.

For example, if C_BASEADDR = 0xE0000000, C_HIGHADDR must be at least = 0xE000003F, when C_IS_DUAL = 0. The address range size when C_IS_DUAL = 0 is (0xE000003F - 0xE0000000) + 1 = 0x40. And C_HIGHADDR must be at least = 0xE000103F, when C_IS_DUAL = 1. The address range when C_IS_DUAL = 1 is (0xE000103F - 0xE0000000) + 1 = 0x1040.

XPS PS2 Controller Parameter - Port Dependencies

The dependencies between the XPS PS2 Controller core design parameters and I/O signals are described in Table 3. In addition, when certain features are parameterized out of the design, the related logic will no longer be a part of the design. The unused input signals and related output signals are set to a specified value.

Table 3: **XPS PS2 Controller Design Parameter - Port Dependencies**

Generic or Port	Name	Affects	Depends	Relationship Description
Design Parameters				
G4	C_SPLB_AWIDTH	P3	-	Affects number of bits in address bus
G5	C_SPLB_DWIDTH	P7,P10, P33	-	Affects number of bits in data bus
G7	C_SPLB_MID_WIDTH	P5	G8	Affects the width of current master identifier signals and depends on $\log_2$(C_SPLB_NUM_MASTERS) with a minimum value of 1
G8	C_SPLB_NUM_MASTERS	P36,P37, P38,P42	-	Affects the width of busy and error signals.
G11	C_IS_DUAL	P45,P46, P48		When C_IS _DUAL is 1, port2 is created
I/O Signals				

Table 3: **XPS PS2 Controller Design Parameter - Port Dependencies *(Cont.)***

Generic or Port	Name	Affects	Depends	Relationship Description
P3	PLB_ABus[0: C_SPLB_AWIDTH - 1]	-	G4	Width varies with the size of the PLB address bus
P5	PLB_masterID[0: C_SPLB_MID_WIDTH - 1]	-	G7	Width varies with the size of the PLB master identifier bus
P7	PLB_BE[0: (C_SPLB_DWIDTH/8) -1]	-	G5	Width varies with the size of the PLB data bus
P10	PLB_wrDBus[0: C_SPLB_DWIDTH - 1]	-	G5	Width varies with the size of the PLB data bus
P32	Sl_rdDBus[0: C_SPLB_DWIDTH - 1]	-	G5	Width varies with the size of the PLB data bus
P36	Sl_MBusy[0: C_SPLB_NUM_ MASTERS - 1]	-	G8	Width varies with the size of the PLB number of masters
P37	Sl_MWrErr[0: C_SPLB_NUM_ MASTERS - 1]	-	G8	Width varies with the size of the PLB number of masters
P38	Sl_MRdErr[0: C_SPLB_NUM_ MASTERS - 1]	-	G8	Width varies with the size of the PLB number of masters
P42	Sl_MIRQ[0: C_SPLB_NUM_MAST ERS - 1]	-	G8	Width varies with the size of the PLB number of masters
P45	PS2_2_DATA	-	G11	Depends on whether the dual channel is enabled
P46	PS2_2_CLK	-	G11	Depends on whether the dual channel is enabled

XPS PS2 Controller Register Descriptions

There are eight internal registers in the XPS PS2 Controller design as shown in Table 4. These registers are implemented in the PS2_REG interface module. The memory map of the XPS PS2 Controller design is determined by setting the C_BASEADDR parameter. The internal registers of the XPS PS2 Controller are at a fixed offset from the base address. The XPS PS2 Controller internal registers and their offset are listed in Table 4.

Table 4: **XPS PS2 Controller Internal Registers**

Register Name	Description	Base Address + Offset (hex)	Access
SRST_1	XPS PS2 Port1 Software Reset Register	C_BASEADDR + 0x0000	Write
STATUS_REG1	XPS PS2 Port1 Status Register	C_BASEADDR + 0x0004	Read
RX1_DATA	XPS PS2 Port1 Receive Data Register	C_BASEADDR + 0x0008	Read
TX1_DATA	XPS PS2 Port1 Transmit Data Register	C_BASEADDR + 0x000C	Write
SRST_2	XPS PS2 Port2 Software Reset Register	C_BASEADDR + 0x1000	Write
STATUS_REG2	XPS PS2 Port2 Status Register	C_BASEADDR + 0x1004	Read
RX2_DATA	XPS PS2 Port2 Receive Data Register	C_BASEADDR + 0x1008	Read
TX2_DATA	XPS PS2 Port2 Transmit Data Register	C_BASEADDR + 0x100C	Write

Notes:
1. Writing into a read-only register or reading a write-only register would generate an error. This would be communicated to the PLB Interface Module. Such a transaction would not be a valid transaction.
2. The default value of all the above mentioned registers is zeros.

Table 5 shows the XPS PS2 Port1 interrupt registers and their offset from the base address of XPS PS2 memory map.

Table 5: **PS2 Port1 Interrupt Registers**

Register Name	Base Address + Offset (hex)	Access
XPS PS2 Port1 Global Interrupt Enable Register (GIE_1)	C_BASEADDR + 0x002c	Read/Write
XPS PS2 Port1 Interrupt Status Register (IPISR_1)	C_BASEADDR + 0x0030	Read/TOW(1)
XPS PS2 Port1 Interrupt Enable Register (IPIER_1)	C_BASEADDR + 0x0038	Read/Write

Notes:
1. TOW = Toggle On Write. Writing a '1' to a bit position within the register causes the corresponding bit position in the register to toggle.

Table 6 shows the XPS PS2 Port2 interrupt registers and their offset from the base address of XPS PS2 memory map.

Table 6: **PS2 Port2 Interrupt Registers**

Register Name	Base Address + Offset (hex)	Access
XPS PS2 Port2 Global Interrupt Enable Register (GIE_2)	C_BASEADDR + 0x102c	Read/Write
XPS PS2 Port2 Interrupt Status Register (IPISR_2)	C_BASEADDR + 0x1030	Read/TOW(1)
XPS PS2 Port2 Interrupt Enable Register (IPIER_2)	C_BASEADDR + 0x1038	Read/Write

Notes:
1. TOW = Toggle On Write. Writing a '1' to a bit position within the register causes the corresponding bit position in the register to toggle.

As mentioned in the above table, the interrupt/status/data registers of PS2 Port1 start at the base address and the interrupt/status/data registers of the PS2 Port2 start at base address + 0x1000.

Depending on whether the parameter C_IS_DUAL is 1 or 0, the status/data registers SRST_2, STATUS_REG2, RX2_DATA and TX2_DATA and the port2 interrupt registers GIE_2, IPISR_2 and IPIER_2 are included or removed respectively.

XPS Portx Software Reset Register (SRST_x)

The XPS PS2 Controller has Internal Software Reset Register (SRST_x) for each of its two ports, resetting the registers independently. The reset register is shown in Figure 4. It is a write only register addressed at an offset 0x0 from base address C_BASEADDR/C_BASEADDR+0x10. The bit definitions of this register are as shown in Table 7.

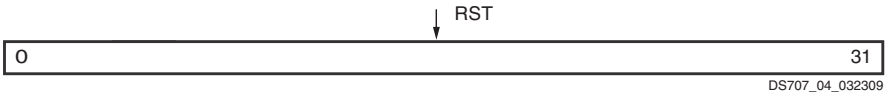

Figure 4: **Software Reset Register (SRST)**

Table 7: **SRST Register Bit Definitions**

Bits	Name	Core Access	Reset Value	Description
0 - 31	SRST_x	Write	N/A	**Software Reset** A write of 0x0000000A causes reset of the XPS PS2 Controller. A write of any other value has undefined effect and returns a bus error. A read of this register returns zero.

XPS PS2 Portx Status Register

The XPS PS2 Portx Status Register indicates the status of the PS2 Portx while transmitting and receiving data packets. Figure 5 and Table 8 give a more detailed description of the status bits which do the same.

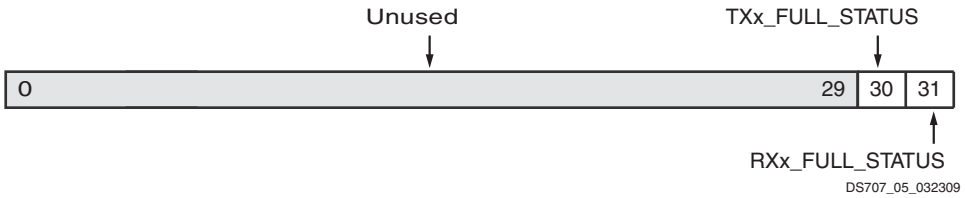

Figure 5: **XPS PS2 Status Register**

Table 8: **XPS PS2 Status Register Description**

Bit(s)	Name	Core Access	Reset Value	Description
0 to 29	Unused	N/A	0	Unused
30	TXx_Full_Status	Read	'0'	PS2 Portx SIE is busy. This field is set when the data to be transmitted is written into the transmit register. This is field is cleared when the transmission is done. The software has no direct write permission to change this field, since this field is set by the state machine in the SIE. However, the software can indirectly do that by using the Soft Reset Register
31	RXx_Full_Status	Read	'0'	This field is set when the data packet is received by the Portx SIE and is waiting for the host to read this data packet. This field is cleared once the data packet is read by the host. The software has no direct write permission to change this field, since this field is set by the state machine in the SIE. However, the software can indirectly do that by using the Soft Reset Register

XPS PS2 Portx Receive Data Register

XPS PS2 Portx Receive Data Register is used to read the data line during device-to-host communication. The device writes the data into this register and the host reads back the data from this register to perform the appropriate functions. Table 9 gives a detailed description of the XPS PS2 Portx Receive Data Register.

The XPS PS2 Portx Receive Data Register is shown in Figure 6

Figure 6: **XPS PS2 Receive Data Register**

Table 9: **XPS PS2 Receive Data Register Description**

Bit(s)	Name	Core Access	Reset Value	Description
0 to 23	Unused	N/A	0	Unused
24 to 31	RXx_DATA	Read	0	PS2 Portx Receive Data from the device to host

XPS PS2 Portx Transmit Data Register

XPS PS2 Portx Transmit Data Register is drive the data line during host-to-device communication. The host writes the data into this register and the SIE transmits the data on the data line for the device to perform the necessary function. Table 10 gives a detailed description of the XPS PS2 Portx Transmit Data Register.

The XPS PS2 Portx Transmit Data Register is shown in Figure 7

Figure 7: **XPS PS2 Transmit Data Register**

Table 10: **XPS PS2 Transmit Data Register Description**

Bit(s)	Name	Core Access	Reset Value	Description
0 to 23	Unused	N/A	0	Unused
24 to 31	TXx_DATA	Write	0	PS2 Portx Transmit Data from the host to device

XPS PS2 Controller Interrupts

The interrupts generated by XPS PS2 Controller are managed by the Interrupt Service Controller (ISC). This unit provides many of the features commonly provided for interrupt handling.To support interrupt capability for the two PS2 ports, the PLB interface module implements the following registers:

- Global Interrupt Enable register (GIE)
- IP Interrupt Enable Register (IP IER)
- IP Interrupt Status Register (IP ISR)

The IP IER implements independent interrupt enable bit for each PS2 port while the Global Interrupt Enable Register provides the master enable/disable for the interrupt output to the processor. The IP ISR implements independent interrupt status bit for each PS2 port. The IP ISR provides Read and Toggle-On-Write access. The Toggle-On-Write mechanism for the IP Interrupt Status Register avoids the requirement on the user interrupt service routine to perform Read-Modify-Write operation to clear the status bit of the interrupt. Read-Modify-Write operations can lead to inadvertent clearing of interrupts captured in the time period between the read and write operations. Please refer to the *Processor IP Reference Guide* under Part 1 for a complete description of the GIE, IPIER and IPISR.

XPS PS2 Portx Global Interrupt Enable Register (GIE_x)

The Device Global Interrupt Enable Register provides the final enable/disable for the interrupt output to the processor and resides in the PLB Interface Module. This is a single bit read/write register as shown in Figure 8. Table 11 shows the GIE bit definitions.

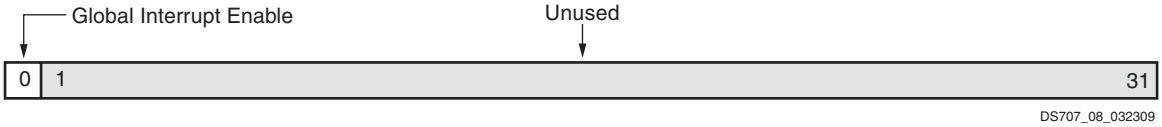

Figure 8: **Device Global Interrupt Enable Register**

Table 11: **Device Global Interrupt Enable Register (GIE) Bit Definitions**

Bit(s)	Name	Core Access	Reset Value	Description
0	Global Interrupt Enable	Read/Write	'0'	Master Enable for routing Device Interrupt to the System Interrupt Controller. '1' = Enabled '0' = Disabled
1 to 31	Unused	N/A	0	Unused

XPS PS2 Portx Interrupt Status and Interrupt Enable Registers

XPS PS2 Portx Interrupt Status Register (IPISR_x) and XPS PS2 Portx Interrupt Enable Register (IPIER_x), respectively. See Figure 9, Table 12 and Table 13.

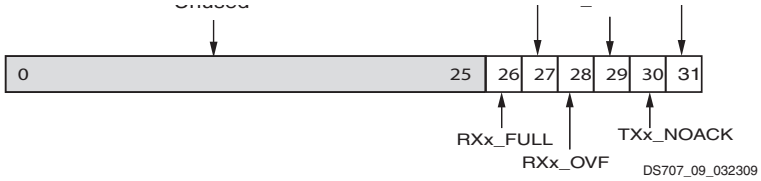

Figure 9: **Interrupt Status and Interrupt Enable Register**

Table 12: **XPS PS2 Portx Interrupt Status Register (IPISR) Bit Definitions**

Bit(s)	Name	Core Access	Reset Value	Description
0 to 25	Unused	N/A	0	Unused
26	RXx_FULL	Read/TOW	'0'	Portx RX Data Register Full This bit is set by the PS2 SIE, when it has received a data packet
27	RXx_ERR	Read/TOW	'0'	Portx RX Data Error This bit is set by the PS2 SIE, when it has received a bad packet
28	RXx_OVF	Read/TOW	'0'	Portx RX Data Register Overflow This bit is set by the PS2 SIE, when a data packet is overwritten before the previous data was read

Table 12: **XPS PS2 Portx Interrupt Status Register (IPISR) Bit Definitions**

Bit(s)	Name	Core Access	Reset Value	Description
29	TXx_ACKF	Read/TOW	'0'	Portx TX Data Acknowledge Received This bit is set by the PS2 SIE, when it has completed the transmission of a packet and received an acknowledgement from the PS2 device
30	TXx_NOACK	Read/TOW	'0'	Portx TX Data Acknowledge not Received This bit is set by the PS2 SIE, when it has completed transmission of a packet and has not received an acknowledgement from the PS2 device
31	WDTx_TOUT	Read/TOW	'0'	Portx WatchDog Timer Timeout This bit is set by the PS2 SIE, when it has not received the clock from the PS2 device while transmission of a packet

Table 13: **XPS PS2 Portx Interrupt Enable Register (IPIER) Bit Definitions**

Bit(s)	Name	Core Access	Reset Value	Description
0 to 25	Unused	N/A	0	Unused
26	RXx_FULL	Read/Write	'0'	Enable/Disable Portx RX Data Register Full Interrupt '1' = Enabled '0' = Disabled (masked)
27	RXx_ERR	Read/Write	'0'	Enable/Disable Portx RX Data Error Interrupt '1' = Enabled '0' = Disabled (masked)
28	RXx_OVF	Read/Write	'0'	Enable/Disable Portx RX Data Register Overflow Interrupt '1' = Enabled '0' = Disabled (masked)

Table 13: **XPS PS2 Portx Interrupt Enable Register (IPIER) Bit Definitions**

Bit(s)	Name	Core Access	Reset Value	Description
29	TXx_ACKF	Read/Write	'0'	Enable/Disable Portx TX Data Acknowledge Received Interrupt '1' = Enabled '0' = Disabled (masked)
30	TXx_NOACK	Read/Write	'0'	Enable/Disable Portx TX Data Acknowledge not Received Interrupt '1' = Enabled '0' = Disabled (masked)
31	WDTx_TOUT	Read/Write	'0'	Enable/Disable Portx Watch Dog Timer Timeout Interrupt '1' = Enabled '0' = Disabled (masked)

Design Implementation

Target Technology

The target technology is an FPGA listed in EDK Supported Device Families.

Device Utilization and Performance Benchmarks

Core Performance

Because the XPS PS2 Controller core will be used with other design modules in the FPGA, the utilization and timing numbers reported in this section are estimates only. When the XPS PS2 Controller core is combined with other designs in the system, the utilization of FPGA resources and timing of the XPS PS2 Controller design will vary from the results reported here.

The XPS PS2 Controller resource utilization for various parameter combinations measured with Virtex-4 as the target device are detailed in Table 14.

Table 14: **Performance and Resource Utilization Benchmarks on Virtex-4 (xc4vfx60-ff672-10)**

Parameter Values	Device Resources			Performance
C_IS_DUAL	Slices	Slice Flip-Flops	LUTs	F_{MAX} (MHz)
0	282	341	455	196
1	477	582	890	186

The XPS PS2 Controller resource utilization for various parameter combinations measured with Virtex-5 as the target device are detailed in Table 15.

Table 15: **Performance and Resource Utilization Benchmarks on Virtex-5 (xc5vlx50t-ff665-1)**

Parameter Values	Device Resources			Performance
C_IS_DUAL	Slices	Slice Flip-Flops	LUTs	F_{MAX} (MHz)
0	127	341	318	220
1	252	582	663	180

The XPS PS2 Controller resource utilization for various parameter combinations measured with Spartan-3 as the target device are detailed in Table 16.

Table 16: **Performance and Resource Utilization Benchmarks on Spartan-3 (xc3sd3400a-cs484-4)**

Parameter Values	Device Resources			Performance
C_IS_DUAL	Slices	Slice Flip-Flops	LUTs	F_{MAX} (MHz)
0	290	358	420	110
1	516	582	807	101

The XPS PS2 Controller resource utilization for various parameter combinations measured with Virtex-6 as the target device are detailed in Table 17.

Table 17: **Performance and Resource Utilization Benchmarks on Virtex-6 (xc6vlx195t-1-ff1156)**

Parameter Values	Device Resources			Performance
C_IS_DUAL	Slices	Slice Flip-Flops	LUTs	F_{MAX} (MHz)
0	161	338	447	224
1	305	558	785	197

The XPS PS2 Controller resource utilization for various parameter combinations measured with Spartan-6 as the target device are detailed in Table 18.

Table 18: **Performance and Resource Utilization Benchmarks on Spartan-6 (xc6slx45-2-fgg484)**

Parameter Values	Device Resources			Performance
C_IS_DUAL	Slices	Slice Flip-Flops	LUTs	F_{MAX} (MHz)
0	160	338	394	122
1	271	558	710	116

System Performance

To measure the system performance (Fmax) of this core, this core was added to a Virtex-4 system, a Virtex-5 system, and a Spartan-3A system as the Device Under Test (DUT) as shown in Figure 10, Figure 11, and Figure 12.

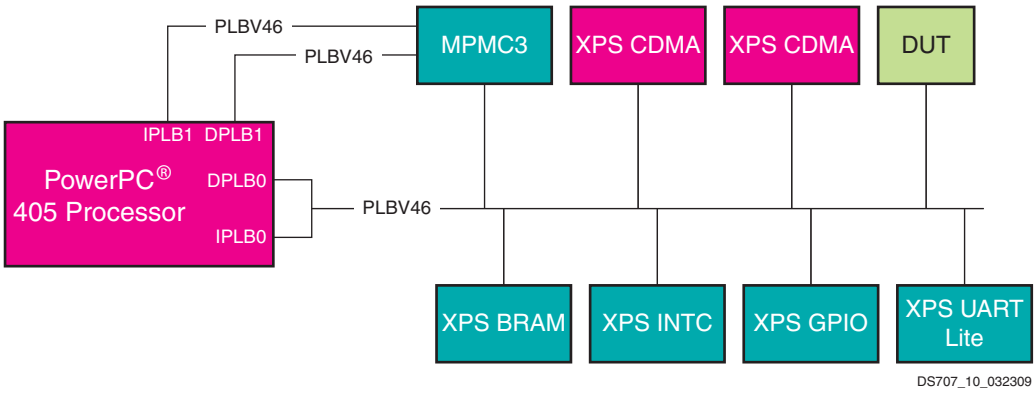

Figure 10: **V4 FX System**

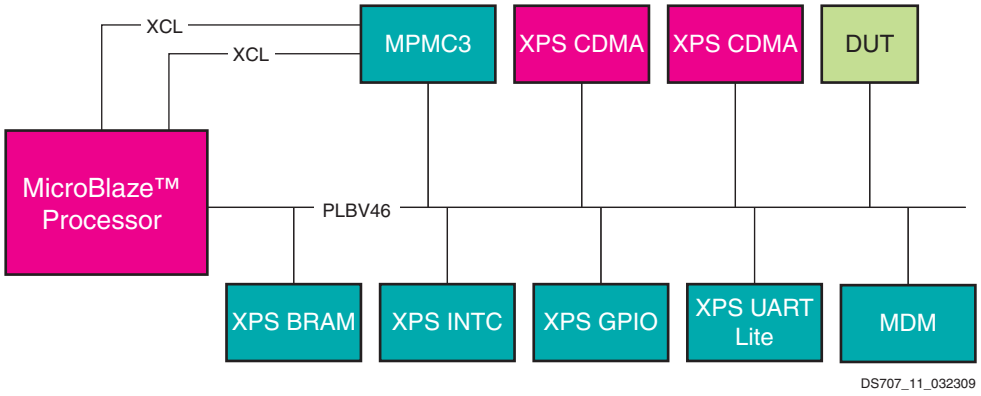

Figure 11: **V5 LX System**

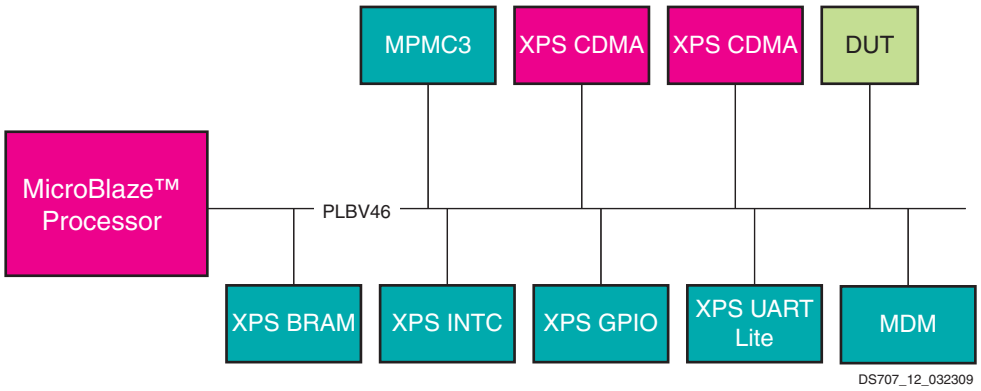

Figure 12: **Spartan-3A System**

The target FPGA was then filled with logic to drive the LUT and BRAM utilization to approximately 70% and the I/O utilization to approximately 80%. Using the default tool options and the slowest speed grade for the target FPGA, the resulting target F_{MAX} numbers are shown in Table 19.

Table 19: **XPS PS2 Controller Core System Performance**

Target FPGA	Target F_{MAX} (MHz)
S3A700 -4	90
V4FX60 -10	100
V5LXT50 -1	120

The target F_{MAX} is influenced by the exact system and is provided for guidance. It is not a guaranteed value across all systems.

Specification Exceptions

N/A

Reference Documents

1. IBM CoreConnect 128-Bit Processor Local Bus, Architectural Specification (v4.6).

Revision History

Date	Version	Revision
04/17/07	1.0	Initial Xilinx release.
7/25/08	1.1	Added QPro Virtex-4 Hi-Rel and QPro Virtex-4 Rad Tolerant support.
4/24/09	1.2	Replaced references to supported device families and tool name(s) with hyperlink to PDF file.
7/20/09	1.3	S6/V6 resource utilization tables added

Notice of Disclaimer