XPS_I2C_SLAVE custom IP core block

Description

The logic block XPS_I2C_SLAVE is a I2C communication core used for I2C communication between Xilinx FPGA's MicroBlaze soft processor and USB FX2 microcontroller. It is usually used for command, settings and status communication. The I2Cserial communication frequency is high speed (400 kHz).

With every Logical Architecture Layer (FPGA image) using MicroBlaze (with interrupt controller xps_intc configured to use xps_i2c_slave_0_IP2INTC_Irpt) and custom USB FX2 microcontroller's firmware:

- XPS_I2C_SLAVE could be used to interface Microblaze's software and the USB FX2 microcontroller's firmware;
- this way a safe I2C bidirectional communication between the FPGA soft processor MicroBlaze and the USB FX2 microcontroller is possible.

With the Referece Architecture Layer (FPGA image) or compatible derived Logical Architecture Layer (FPGA image) using MicroBlaze and original (Trenz Electronic) USB FX2 microcontroller's firmware:

- XPS_I2C_SLAVE could be used for low speed bidirectional communication between the FPGA and a host computer (through USB FX2 microcontroller).
- this way a safe USB bidirectional communication (through USB connection and USB FX2 microcontroller's I2C connection) between the FPGA soft processor MicroBlaze and host computer is possible.

? Unknown Attachment

System integration block scheme

The XPS_I2C_SLAVE has 2 bus interfaces:

1. Slave Phillips® I2C bidirectional serial interface.
2. Xilinx PLBv4.6 created with IPIF wizard for access to 6 32-bit registers

system.mhs extract

```
PORT USB_INT = USB_INT0   //PA0/INT0 PIN         //USB_INT/INT0 of the image above and below
PORT USB_SCL = USB_SCL                          //I2C[0] of the image above and USB_SCL of the image below
PORT USB_SDA = USB_SDA                          //I2C[1] of the image below and USB_SDA of the image below
PORT IP2INTC_Irpt = xps_i2c_slave_0_IP2INTC_Irpt //IP2INTC_Irpt of the image above and below
```

For example, see https://github.com/Trenz-Electronic/TE063X-Reference-Designs/blob/master/reference-TE0630/system.mhs.

? Unknown Attachment

Peripheral internal structure block scheme: XPS_I2C_SLAVE internals

Programming model

With every Logical Architecture Layer (FPGA image) using MicroBlaze (with interrupt controller xps_intc configured to use xps_i2c_slave_0_IP2INTC_Irpt) and custom USB FX2 microcontroller's firmware:

- when the host computer's SW or the FX2 microcontroller's FW sends a MB Command to the MicroBlaze (MB) soft embedded processor (FX22MB_REG0 will be written as a result), the signal xps_i2c_slave_0_IP2INTC_Irpt is rised;
- when the FPGA's microprocessor (MicroBlaze) receives an interrupt (IP2INTC_Irpt) it should read all FX2MB_REGs for new instructions (aka original or custom MB Command) => the user should write an interupt handler (that actually execute the delivered MB Command) using i2c_slave_int_handler() function in interrupt.c as start point;
- when MicroBlaze's software wants to send information to USB FX2 microcontroller, it should write MB2FX2_REGs (write MB2FX2_REG0 automatically triggers USB_INT);
- we recommend that the last write is to MB2FX2_REG0 since it triggers USB_INT (PA0/INT0 pin);
- when the USB_INT is triggered the FX2 microcontroller's firmware could automatically, or not (it depends on custom or reference FX2 microcontroller's firmware), reads all registers (or a programmed number of bytes);
- this way a safe I2C bidirectional communication between the FPGA soft processor MicroBlaze and the USB FX2 microcontroller is possible;

- the connection between FPGA's MicroBlaze and host computer (through FX2 microcontroller) is custom FPGA image, USB FX2 microcontroller's firmware and host computer's software dependent => the two connection type (A and/or B) used in reference design could be used as start point or another type could be created;

With the Referece Architecture Layer (FPGA image) or compatible derived Logical Architecture Layer (FPGA image) using MicroBlaze and original (Trenz Electronic) USB FX2 microcontroller's firmware:

- connection type A, host computer send a
  - SET_INTERRUPT command;
  - MicroBlaze API Command (MB Command) using the I2C_WRITE command;
  - GET_INTERRUPT command to poll FX2 microcontroller for interrupt data (MB2FX2_REGs) and USB_INT status bit.
- connection type B, host computer send a
  - MicroBlaze API Command (MB Command) using the I2C_WRITE command => the FX2 microcontroller's FW dispatch the MB Command to FPGA's MicroBlaze and it doesn't expect any reply => FX2 microcontroller's firmware is not enabled (sts_int_auto_configured=0) to read MB2FX2_REGs and host computer's SW doesn't poll FX2 microcontroller for autoresponse data (MB2FX2_REGs) and USB_INT status bit.

This way a safe bidirectional communication (through USB connection and USB FX2 microcontroller) between the FPGA microprocessor and computer is possible.

command Byte	Value	Description
0 SW:command[0] FW:EP1OUTBUF[0]	0xAD	I2C_WRITE USB FX2 API command ID
1 SW:command[1] FW:EP1OUTBUF[1]	0x3F	I2C Address FX2_Parameters.MB_I2C_ADRESS=0x3F of host software enum __xdata BYTE iar_adress = 0x3F; of firmware te_api.c iar_adress = EP1OUTBUF[1]; of firmware te_api.c C_I2C_ADDRESS must be set properly for an I2C_SLAVE to be recognized by FX2. Address 63 (0x3F) is used in all reference designs.
2 SW:command[2] FW:EP1OUTBUF[2]	0x0C (12)	FX2_Parameters.I2C_BYTES=0x0C of host software enum __xdata BYTE iar_count = 12; of firmware te_api.c iar_count = EP1OUTBUF[2]; of firmware te_api.c Number of bytes to write (max 32)
3 SW:command[3] FW:EP1OUTBUF[3]	0x00	-
4 SW:command[4] FW:EP1OUTBUF[4]	0x00	-
5 SW:command[5] FW:EP1OUTBUF[5]	0x00	-
6 SW:command[6] FW:EP1OUTBUF[6]	Command2MB	MB_Command ID to send to the MicroBlaze

From 7 to 63	-	Not used

MB_Command Packet Layout with the SW API TE_USB_FX2_SendCommand(..., command, …).

Connection type A.

Command (aka host computer software's MB command to FPGA's MB2FX2_REGs) and reply (aka FPGA's MB2FX2_REGs to FX2 microcontroller firmware's autoresponse bytes array and host computer software's reply bytes array)

- The host computer's software enable (sts_int_auto_configured=1) FX2 microcontroller's firmware to read MB2FX2_REGs; FX2 microcontroller's firmware reading of MB2FX2_REGs is enabled by host computer's software C++ TE_USB_FX2_SendCommand(...,command,...) or C# TE_USB_FX2_SendCommand(...,command,...) used with command[0]=SET_INTERRUPT command; this command sets address and number of bytes to read from the I2C bus when an interrupt request (USB_INT) is received (in reference FX2 firmware v3.02 or v3.01, the interrupt request USB_INT is NOT served by an interrupt but by a normal function "Interrupt Pin polling" IntPinPool() in FW v3.01 or int_pin_pool() in FW v3.02 running in the superloop while(1) of fw.c).
- The host computer's software C++ TE_USB_FX2_SendCommand(...,command,...) or C# TE_USB_FX2_SendCommand(...,command,...) used with command[0]=I2C_WRITE command (this command sets address and number of bytes to read from the I2C bus when an interrupt request (USB_INT) is received) and command[6] = MB_Command.
- The prevoius TE_USB_FX2_SendCommand(...,command) sends a MB Command to the MicroBlaze (MB) soft embedded processor (FX22MB_REG0 will be written as a result) => the signal xps_i2c_slave_0_IP2INTC_Irpt is rised because FX22MB_REG0 is written.
- When the FPGA's microprocessor (MicroBlaze) receives an interrupt request (IP2INTC_Irpt) it should read all FX22MB_REGs for new instructions (aka MB Command) => If the reference architecture test program demo.elf is running, a "FX2 interrupt handler" (i2c_slave_int_handler () function in interrupt.c) is called to handle the signal xps_i2c_slave_0_IP2INTC_Irpt. The i2c_slave_int_handler() function actually execute the delivered MB Command;
- When MicroBlaze's software wants to send information to the host computer (through USB FX2 microcontroller), it should write MB2FX2_REGs.
- We recommend that the last write is to MB2FX2_REG0 since it triggers USB_INT (PA0/INT0 pin).
- When the USB_INT is triggered (and sts_int_auto_configured=1) the FX2 microcontroller's firmware automatically reads all registers (or a programmed number of bytes). The host computer polls FX2 microcontroller for autoresponse (aka interrupt) data and USB_INT status bit; to do a single poll the C++ TE_USB_FX2_SendCommand(...,command,...) or C# TE_USB_FX2_SendCommand(...,command,...) should be used with c ommand[0]=GET_INTERRUPT command.
- This way a safe USB bidirectional communication (through USB connection and USB FX2 microcontroller I2C connection) between the FPGA soft processor MicroBlaze and host computer is possible.

Example of type A connection.

FX22MB_REG0_GETVERSION command procedure.

1. The host computer's software enable ((SW SendCommand(...,command,...) with command[0]=SET_INTERRUPT=0xB0 => FW CMD_SET_AUTORESPONSE=0xB0 =>sts_int_auto_configured = 1)) FX2 microcontroller's firmware to read MB2FX2_REGs (autoresponse to interrupt request (USB_INT) should be preconfigured (sts_int_auto_configured = 1) by the user) (//"AUTORESPONSE: 1st section of code to run" in the firmware code te_api.c).
2. Send the MB Command FX22MB_REG0_GETVERSION (FX22MB_REG00 will be written as a result).
    a. The MicroBlaze execute this MB Command and writes data (Version Information of Reference Architecture running on the FPGA) to MB2FX2_REG0.
    b. When MB write data to MB2FX2_REG0, the interrupt request pin INT0 (aka FPGA_INT0 in firmware files) is rised. This pin is connected to PA0/INT0 pin of FX2 microcontroller.
    c. When the FX2 microcontroller's firmware read the rise of pin INT0 (=USB_INT=1 because MicroBlaze writes data to MB2FX2_REG0) it set the firmware variable FPGA_INT0 to 1.
    d. If an autoresponse to interrupt request (USB_INT) is preconfigured (sts_int_auto_configured = 1) and FPGA_INT0=USB_INT=1, FX2 microcontroller firmware reads all MB2FX2 registers. The registers value are copied in the byte array auto_response_data by the "Interru pt Pin polling" int_pin_pool() firmware function (//"AUTORESPONSE: 2nd section to run" in the firmware code te_api.c).
3. After this the host computer's software should get the autoresponse value on the host (SW SendCommand(...,command,...) with command[0] =GET_INTERRUPT=0xB1 => FW CMD_SET_AUTORESPONSE=0xB1 => EP1INBUF[0] = iar_int_idx; for(i = 0; i < 32; i++) EP1INBUF[i+1] = auto_response_data[i];) (//"AUTORESPONSE: 3rd section to run" in the firmware code te_api.c).

Connection type B.

Command (aka host computer software's MB command to FPGA's MB2FX2_REGs) with no reply.

- The host computer's software C++ TE_USB_FX2_SendCommand(...,command,...) or C# TE_USB_FX2_SendCommand(...,command,...) is executed with command[0]=I2C_WRITE command (this command sets address and number of bytes to read from the I2C bus when an interrupt request (USB_INT) is received) and command[6] = MB_Command;

- The prevoius TE_USB_FX2_SendCommand(...,command) sends a MB Command to the MicroBlaze (MB) soft embedded processor (FX22MB_REG0 will be written as a result) => the interrupt request xps_i2c_slave_0_IP2INTC_Irpt is rised because FX22MB_REG0 is written;
- When the FPGA's microprocessor (MicroBlaze) receives an interrupt request (IP2INTC_Irpt) it should read all FX2MB_REGs for new instructions (aka MB Command) => If the reference architecture test program demo.elf is running, a "FX2 interrupt handler" (i2c_slave_int_handler() function in interrupt.c) is called to handle the signal xps_i2c_slave_0_IP2INTC_Irpt. The i2c_slave_int_handler() function actually execute the delivered MB Command;
- This way a safe dispatching of a MB Command (from host computer's SW through USB connection and USB FX2 microcontroller I2C connection) to FPGA soft processor is possible.

For using the software header read comments in:

- #project#(or IP reposit.)\drivers\XPS_I2C_SLAVE_v1_00_a\src\XPS_I2C_SLAVE.h

XPS_I2C_SLAVE Core VHDL Design Parameters

Feature/Description	Parameter Name	Allowable Values	Default Value	VHDL Type
System Parameters				
target FPGA family s	C_FAMILY	partan3, spartan3e, spartan3a, spartan3adsp, spartan3an, virtex2p, virtex4, qvirtex4, qrvirtex4, virtex5	virtex5	string
PLB Parameters				
PLB base address	C_BASEADDR	Valid Address	None	std_logic_vector
PLB high address	C_HIGHADDR	Valid Address	None	std_logic_vector
PLB least significant address bus width	C_SPLB_AWIDT	32	32	integer
PLB data width	C_SPLB_DWIDTH	32, 64, 128	32	integer
Shared bus topology	C_SPLB_P2P	0 = Shared bus topology	0	integer
PLB master ID bus Width	C_SPLB_MID_WIDTH	log2(C_SPLB_NUM_MASTERS) with a minimum value of	1	integer
Number of PLB masters	C_SPLB_NUM_MASTER	1-16	1	integer
Width of the slave data bus	C_SPLB_NATIVE_DWIDT	32	32	integer
Burst support	C_SPLB_SUPPORT_BURSTS	0 = No burst support	0	integer
XPS_I2C_SLAVE Parameters				
I2C slave address (1)	C_I2C_ADDRESS	0-127	63	integer
Number of bytes to trigger IP2INTC_Irpt (2)	C_MB_INT_BYTE	1-12	12	integer

XPS_I2C_SLAVE Core VHDL Design Parameters

(1) C_I2C_ADDRESS must be set properly for an I2C_SLAVE to be recognized by FX2. Address 63 (0x3F) is used in all reference designs.
(2) C_MB_INT_BYTES can be less than 12 to speed up I2C communication by transferring less information. On the other hand, since the USB latency is high overall speed would not be increased much.
The transactions of these I2C connections are usually 12 bytes long (FX2_Parameters.I2C_BYTES=0x0C of host software enum; __xdata BYTE iar_count = 12 of firmware te_api.c; iar_count = EP1OUTBUF[2] of firmware te_api.c).

XPS_I2C_SLAVE Core I/O Signals

Name	Interface	I/O	Initial State	Description

ChipScope[0:31]	-	O	-	Debug port
USB_IFCLK	-	I	-	USB 48 MHz clock
USB_INT	pin PA/INT0 of FX2 micontroller and FPGA chip	O	0	USB Interrupt request: interrupt request to USB FX2 microcontroller. In control of pin PA0/INT0, it is stored in FX2 USB microcontroller as FPGA_INT0 firmware variable. When the FX2 microcontroller's firmware read the rise of pin INT0 (=1 because MicroBlaze writes data to MB2FX2_REG0) it set the firmware variable FPGA_INT0 to 1.
USB_SCL	-	I	-	USB I2C serial clock
USB_SDA	-	I/O	-	USB I2C serial data I2Cserial communication frequency is high speed (400 kHz).
IP2INTC_Irpt	xps_intc module of MicroBlaze	O	0	MB interrupt request: interrupt request to MicroBlaze interrupt controller (xps_intc module). xps_i2c_slave_0_IP2INTC_Irpt signal. When the host sends a MB Command to the MicroBlaze (MB) soft embedded processor (FX22MB register 0 will be written as a result), the signal xps_i2c_slave_0_IP2INTC_Irpt is rised. If the reference architecture test program demo.elf is running a "FX2 interrupt handler" (i2c_slave_int_handler() function in interrupt.c) is called to handle the signal xps_i2c_slave_0_IP2INTC_Irpt. The i2c_slave_int_handler() function actually execute the delivered MB Command.
OTHERS ARE PLBv4.6 SIGNALS	PLBv4.6			

XPS_I2C_SLAVE I/O Signal Descriptions

XPS_I2C_SLAVE Core Registers

The logic block XPS_I2C_SLAVE has access to MicroBlaze functionality through a 6 × 32-bit memory mapped registers (3 for reading and 3 for writing, for a total of 12 bytes for reading and 12 bytes for writing) attached to PLBv4.6 bus:

- 3 for the host computer's SW(or FX2 microcontroller's FW) => FPGA communication (FX22MB registers; FX22MB_REG0 is fundamental for Micro Blaze API Commands (MB Commands));
- 3 for the software runnig on the Xilinx FPGA's MicroBlaze => host communication (MB2FX2 registers; how and when these registers are read back by FX2 microcontroller is FX2's firmware dependent).

The transactions of these connections are usually 12 bytes long (FX2_Parameters.I2C_BYTES=0x0C of host software enum; __xdata BYTE iar_count = 12 of firmware te_api.c; iar_count = EP1OUTBUF[2] of firmware te_api.c).

When an FPGA writes a word to the first register an interrupt request to the FX2 microcontroller is triggered/rised (USB_INT, pin INT0 is rised) . This pin INT0 is connected to PA0/INT0 pin of FX2 microcontroller. When an interrupt request is triggered the FX2 microcontroller automatically (a custom firmware that serves the interrupt request USB_INT with an ISR) or not (the reference firmware that use a not always enabled "Interrupt Pin polling" function in the while(1) superloop) reads the programmed number of bytes (usually 12) from the XPS_I2C_SLAVE MB2FX2 registers (MB2FX2_REGs). If the current Trenz Electronic reference FX2 microcontroller's firmware is used, the register value are "automatically" read if a autoresponse to interrupt request (USB_INT) is set ("Interrupt Pin polling" function enabled by an autoresponse flag setted by SET_INTERRUPT command). Otherwise the registers value are not automatically readed by FX2 microcontroller's firmware.

When the FX2 writes all 12 bytes to the FPGA registers (FX22MB_REGs) the Microblaze receives an interrupt (xps_i2c_slave_0_IP2INTC_Irpt) to know when the new data (MB Command for reference architecture case) was received. When the host sends a MB Command to the MicroBlaze (MB) soft embedded processor (FX22MB_REG0 will be written as a result), a MicroBlaze's interrupt (xps_i2c_slave_0_IP2INTC_Irpt) is rised and a FX2 interrupt handler (i2c_slave_int_handler() function in interrupt.c running on MicroBlaze) is called to actually execute the delivered MB Command.

Base Address + Offset (hex)	Register Name	Access Type	Default Value (hex)	Description
XPS FX2 IP Core Grouping				
C_BASEADDR + 00	MB2FX2_REG0	R/W	0x00000000	Microblaze to FX2 register 0

C_BASEADDR + 04	MB2FX2_REG1	R/W	0x00000000	Microblaze to FX2 register 1
C_BASEADDR + 08	MB2FX2_REG2	R/W	0x00000000	Microblaze to FX2 register 2
C_BASEADDR + 0C	FX2MB_REG0	Read	0x00000000	FX2 to Microblaze register 0
C_BASEADDR + 10	FX2MB_REG1	Read	0x00000000	FX2 to Microblaze register 1
C_BASEADDR + 14	FX2MB_REG2	Read	0x00000000	FX2 to Microblaze register 2

Details of XPS_I2C_SLAVE Core Registers

Microblaze to FX2 register 0 (MB2FX2_REG0)

A single bit write to this register triggers interrupt to FX2 microcontroller (USB_INT => pin PA0/INT0 => FPGA_INT0 firmware variable).

? Unknown Attachment

PC side: STATUS REGISTER

Microblaze to FX2 register 1 (MB2FX2_REG1)

? Unknown Attachment

PC side: STATUS COMMAND

Microblaze to FX2 register 2 (MB2FX2_REG2)

? Unknown Attachment

PC side: STATUS DATA

FX2 to Microblaze register 0 (FX2MB_REG0)

When FX2 puts a last byte to this register an interrupt is triggered to microprocessor (IP2INTC_Irpt) if C_MB_INT_BYTES is set to 4.

? Unknown Attachment

PC side: CONTROL REGISTER

FX2 to Microblaze register 1 (FX2MB_REG1)

When FX2 puts a last byte to this register an interrupt is triggered to microprocessor (IP2INTC_Irpt) if C_MB_INT_BYTES is set to 8.

? Unknown Attachment

PC side: CONTROL COMMAND

FX2 to Microblaze register 2 (FX2MB_REG2)

When FX2 puts a last byte to this register an interrupt is triggered to microprocessor (IP2INTC_Irpt) if C_MB_INT_BYTES is set to 12.

? Unknown Attachment

PC side: CONTROL DATA