UART to Master AXI3 Lite interface documentation

October 23, 2015

Contents

Introduction **ix**

1 User guide **1**
1.1 Interface 1
1.2 Internal registers 1
1.3 Functional description 2

2 Application notes **5**
2.1 Introduction 5
2.2 Content of the archive 6
2.3 Installing 7
2.4 Running the example 7

List of Tables

1 Revision history ix

1.1 UART2MAXILITE input and output ports 1
1.2 UART to Master AXI3 Lite interface registers table 2
1.3 Commands from UART 3
1.4 Status characters 3

List of Figures

1.1 UART2MAXILITE 4

2.1 UART2MAXILITE in ZedBoard 5

Introduction

This document describes UART2MAXILITE, the UART to Master AXI3 Lite interface of the SecBus project. Chapter 1 is the user guide and presents the module on a purely functional point of view. Chapter 2 gives examples of use of the interface on the ZedBoard[5], a prototyping board based on the Xilinx Zynq core[4].

Revision history

Date	Version	Revision
2015-10-09	1.0	Initial release.
2015-10-23	1.1	Add STATUS register.

Table 1: Revision history

Chapter 1

User guide

Introduction

This chapter presents the functional view of UART2MAXILITE, a UART to Master AXI3 Lite interface. The reader interested in using the interface from a pure functional perspective will find the description of the internal registers and their role.

1.1 Interface

UART2MAXILITE is a hardware component with a system clock and reset, two master AXI ports (`UART_AXI` and `MEM_AXI`), an 8-bits general purpose input (`GPI`), an 8 bits general purpose output (`GPO`) and a synchronous active high reset (`SRST`). Table 1.1 lists the input and output ports.

Name	Direction	Bit-width		Description
		Data	Address	
`ACLK`	input	1	-	System clock
`ARESETN`	input	1	-	System reset
`SRST`	input	1	-	Registers synchronous reset
`UART_AXI`	-	32	12	AXI3 lite master interface with UART
`MEM_AXI`	-	32	32	AXI3 lite master interface with memory mapped device
`GPI`	input	8	-	General Purpose Input
`GPO`	output	8	-	General Purpose Output

Table 1.1: UART2MAXILITE input and output ports

1.2 Internal registers

UART2MAXILITE embeds a set of 8-bits internal registers. Table 1.2 lists the registers. Their default value is defined in the `bitfield_pkg.vhd` package. All registers are initialized to their default value when the `SRST` synchronous reset is asserted high

or when the system reset is asserted low. The difference between the two resets is that the system reset also resets other registers (state registers of state machines...).

Table 1.2: UART to Master AXI3 Lite interface registers table

Name	Address	Dir.	Description
a0	0x0	rw	Byte 0 (least significant) of address
a1	0x1	rw	Byte 1 of address
a2	0x2	rw	Byte 2 of address
a3	0x3	rw	Byte 3 (Most significant) of data
d0	0x4	rw	Byte 0 (least significant) of data
d1	0x5	rw	Byte 1 of data
d2	0x6	rw	Byte 2 of data
d3	0x7	rw	Byte 3 (Most significant) of data
status	0x8	r	Copy of the status register of the UART. Error and TX FIFO full bits are sticky.

The STATUS register is a copy of the UART status register that UART2MAXILITE periodically reads to detect incomming characters from the UART. The 3 bits corresponding to the parity, frame and overrun errors, plus the bit corresponding to TX FIFO full are sticky: once set they can be deasserted only by the system or the synchronous reset. The content of this register cannot be sent to the UART but it can however be sent to the GPO if GPI is set to its index.

1.3 Functional description

UART2MAXILITE has two 32-bits master AXI3 lite interfaces. UART_AXI connects to the Xilinx UART from which characters are received and sent. MEM_AXI connects to a memory mapped device (MEM) and is used to perform read write accesses. UART2MAXILITE does not use interrupts to detect incoming characters from the UART. Instead, it continuously polls the UART status register until the RX FIFO Valid Data flag indicates that a character is available in the receive FIFO.

The characters received from the UART are commands. A command is one or two-characters long, denoted C0 and C1 in the following. The commands and their binary encoding is summarized in table1.3 where x represents a bit which value is ignored while 0,1,W,A are meaningful bit values.

The address of MEM read and MEM write commands is given by registers a0 (least significant byte), a1, a2 and a3 (most significant byte). The data returned by a MEM read command is stored in registers d0 (least significant byte), d1, d2, d3 (most significant byte). The data written by a MEM write command is given by registers d0 (least significant byte), d1, d2, d3 (most significant byte).

After completion of a MEM read or MEM write command a status character is sent to the UART. Table 1.4 lists the status characters.

The GPI general purpose input selects the value sent to the GPO general purpose output: if GPI is a valid register address, the content of the register drives GPO. Else, GPO is set to "00ASLBRT" where:

- A is the active low system reset
- S is the active high synchronous reset

C0	Command
xx00xxxx	MEM read: launch a read access on AXI interface MEM_AXI.
xx01WWWW	MEM write: launch a write access on AXI interface MEM_AXI. WWWW is the 4-bits byte enable (field WSTRB of the AXI write request).
xx10xAAA	Register read: send to UART the content of register at address AAA.
xx11xAAA	Register write: read next character (C1) from UART and store it in register at address AAA.

Table 1.3: Commands from UART

Value	Description
0	OKAY, last MEM_AXI request response was OKAY
2	SLVERR, last MEM_AXI request response was SLVERR
3	DECERR, last MEM_AXI request response was DECERR

Table 1.4: Status characters

- L is the active low local reset (combination of system and synchronous reset)
- B blinks every 2^{25} clock cycles
- R is the registered RX line of the UART
- T is the registered TX line of the UART

Figure 1.1 represents the UART to Master AXI3 Lite interface.

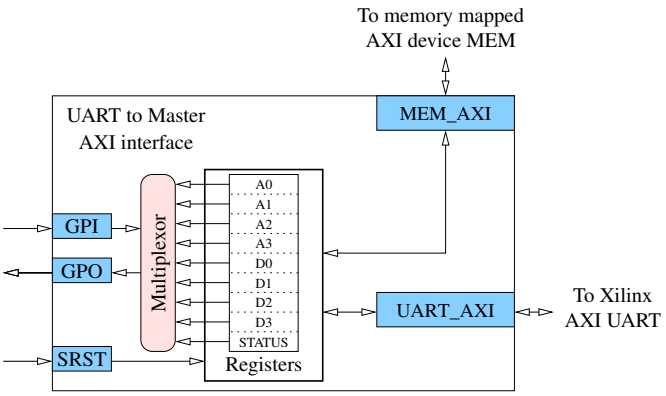

Figure 1.1: UART2MAXILITE

Chapter 2

Application notes

2.1 Introduction

This application note proposes examples of use of UART2MAXILITE on the ZedBoard[5], a prototyping board based on the Xilinx Zynq core[4].

Figure 2.1 represents UART2MAXILITE in its environment.

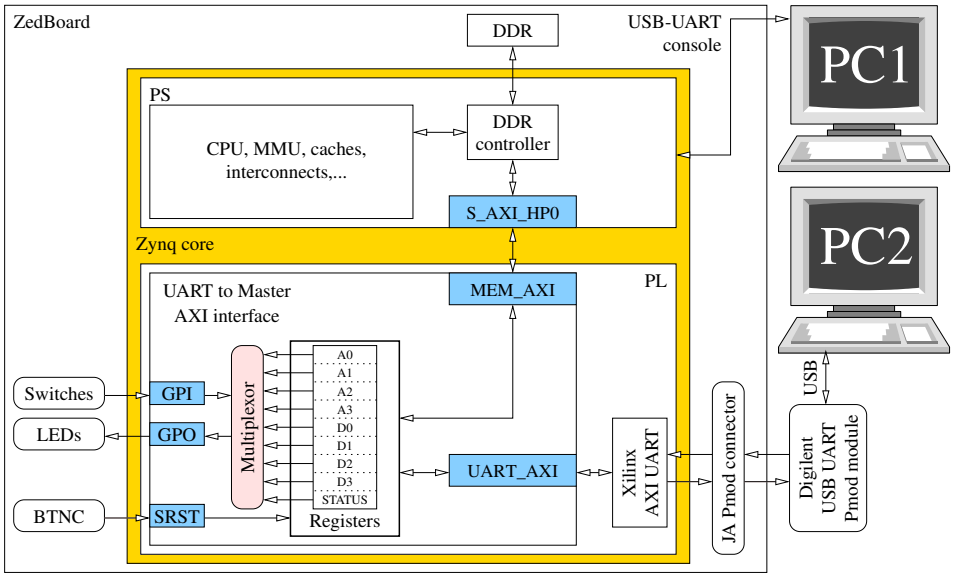

Figure 2.1: UART2MAXILITE in ZedBoard

A PC is attached to the Programmable Logic (PL) of the Zynq core of the ZedBoard through a USB-A / micro-USB cable connected to a Digilent USB-to-UART Pmod module. The Pmod module is plugged to the JA Pmod connector (top row of the 12-pins connector) of the board. The PL is configured with a Xilinx AXI lite UART and UART2MAXILITE. The serial interface of the Xilinx AXI lite UART core is connected to the Pmod module. Its slave AXI interface is connected to the UART_AXI master AXI interface of UART2MAXILITE. The other master AXI in-

terface of UART2MAXILITE, MEM_AXI, is connected to the DDR controller of the Zynq. UART2MAXILITE repeatedly reads the characters received from the attached PC, interprets them as commands, performs the requested memory accesses using MEM_AXI and sends the result back to the PC through the UART and the Pmod module. This set up allows to access the complete DDR of the ZedBoard[1] from a laptop or desktop PC.

The provided software example relies on `libftdi`[2] (the Digilent USB-to-UART Pmod module is based on the FT232RQ chip from FTDIChip [1]).

The example has been tested with:

- Xilinx tools: Vivado 2015.3
- Hardware:
  - Digilent ZedBoard revision C.
  - Digilent Pmod USB UART kit 410-212
- Software:
  - laptop running Debian jessie
  - libftdi-dev 0.20-2

2.2 Content of the archive

```
C/ ..................... example software applications,
                         based on libftdi
    data.txt ........... text file used by 'make check'
                         target
    Makefile ........... type 'make' for a list of
                         available targets
    u2m_f2m.c .......... transfer file to memory through
                         UART
    u2m_lib.c .......... utility functions
    u2m_lib.h .......... header file
    u2m_m2f.c .......... transfer memory to file through
                         UART
    u2m_read.c ......... read 32 bits word from memory and
                         print through UART
    u2m_write.c ........ write 32 bits word to memory
                         through UART
COPYING ................ licence (English)
COPYING-FR ............. license (français)
Makefile ............... type 'make' for a list of
                         available targets
README ................. short description
syn.tcl ................ synthesis script for Xilinx Vivado
uart2maxilite.pdf ...... documentation
axi_pkg.vhd ............ utility VHDL package: AXI protocol
bitfield_pkg.vhd ....... utility VHDL package: interface
```

[1] Except the first MB for good technical reasons - please see the Xilinx Zynq Technical Reference Manual

```
                          registers
global.vhd ............. utility VHDL package: misc
                          functions
uart2maxilite_pkg.vhd .. utility VHDL package: constants,
                          commands...
uart2maxilite.vhd ...... VHDL top level
```

2.3 Installing

First unpack the archive and synthesize the example design:

```
$ tar xf uart2maxilite.tgz
$ cd uart2maxilite
$ make syn
```

Prepare a SD card with a boot image containing a First Stage Boot Loader (FSBL), the bitstream of the example design and a U-Boot ELF[2]. Add a Linux kernel, a device tree blob and a root filesystem.

Alternately, use the SD card archive available on the SecBus website[3]. It already contains a boot image, a Linux kernel, a device tree blob and a minimal root filesystem.

Insert the SD card in its slot on the ZedBoard and attach a PC (`PC1` on figure 2.1), to the USB UART connector of the ZedBoard (near the JTAG connector). Copy the provided software source (sub-directory `C`) on a second PC (`PC2` on figure 2.1)[3]. If needed install the `libftdi` library on `PC2` and compile the applications:

```
PC2> cd C
PC2> make all
```

2.4 Running the example

Plug the Digilent USB-UART Pmod module in the top row of the JA Pmod connector of the ZedBoard (near the switches). Attach `PC2` to the Pmod module.

Launch a serial console (`minicom`, `cu`, `putty`, `screen`...) on `PC1`, power up the board and wait until the Linux kernel boots. The switches and the LEDs can be used to observe the activity of the PL, as explained in chapter 1.

You are ready to access the DDR from `PC2`, through UART2MAXILITE: identify a physical memory area in which you can read and write without interfering with the running software (e.g. `[0x1000_0000...0x1000_1000[`) and load it with the content of the `data.txt` example text file:

```
PC2> cd C
PC2> head -1 data.txt
"La Cryptographie Militaire", Auguste Kerckhoffs...
PC2> ./u2m_f2m 0x10000000 data.txt
```

Check that the load worked from the console on `PC1` using, for instance, the BusyBox `devmem` applet:

[2] Please consult the Xilinx documentation if needed

[3] Of course, the same PC can be used, provided it has two USB ports

```
# devmem 0x1000000 64
0x7079724320614C22 ("La Cryp)
```

From `PC2` read back the content of the same memory area, store it in a second file and compare with the provide example text file:

```
PC2> cd C
PC2> wc -m data.txt
672
PC2> ./u2m_m2f 0x10000000 672 data2.txt
PC2> head -1 data2.txt
"La Cryptographie Militaire", Auguste Kerckhoffs...
PC2> diff data.txt data2.txt
PC2>
```

Bibliography

[1] FTDI Chip: http://www.ftdichip.com/.

[2] libFTDI - FTDI USB driver with bitbang mode: https://www.intra2net.com/en/developer/libftdi/.

[3] Secbus, a hardware / software architecture protecting the external memories of an soc: https://secbus.telecom-paristech.fr/.

[4] Xilinx all programmable socs: http://www.xilinx.com/products/silicon-devices/soc.html.

[5] Zedboard community-based web site: http://zedboard.org/.