Reference System: OPB CAN Controller

XAPP913 (v1.0) February 10, 2006

Abstract

This application note describes a reference system that tests the operation of the OPB CAN Core in Loopback mode. The reference system contains four cores. This application note describes the building of a system containing the OPB CAN, the port connections between different cores and the clocking for the OPB CAN and OPB DDR cores. A basic description of the software application provided with the reference system is also given.

This reference system is targeted for the Xilinx SP305 Rev B board.

Included Systems

Included with this application note is one reference system:

- www.xilinx.com/bvdocs/appnotes/xapp913.zip

Introduction

The Controller Area Network (CAN) is a serial communications protocol used to efficiently support distributed real-time control. In the normal operation mode, two CAN controllers communicate with each other. The loopback mode is a diagnostic mode, wherein the CAN core does not participate in bus communication and transmits recessive bits on the CAN bus. The CAN core transmits messages internally and receives the same messages. This application note illustrates the setting up of a system with the OPB CAN core operating in the loopback mode. The OPB CAN Core transmits a message (Standard Data Frame) and the received message is compared with the transmitted message to confirm the basic transmit - receive functionality of the OPB CAN core. Acceptance filters are used to selectively filter messages that are received by a CAN node. In this system, a single acceptance filter has been used and is configured to filter the transmitted identifier, so that it is received by the CAN node.

Reference System Specifics

This reference system is built on a Xilinx Spartan-3™ SP305 Rev B board. The components of this system are the MicroBlaze™ processor, an on-chip hardware debug module, a 32Kb Block RAM, the OPB DDR SDRAM memory controller, 2 DCMs for the DDR controller (one of which also generates the 50MHz OPB CLK), the OPB UART16550 core, the OPB Interrupt Controller core and the OPB CAN core. The system uses an external crystal oscillator for the generation of the CAN CLK. The address map for this system is given in Table 1. The block diagram of this system can be seen in Figure 1.

Block Diagram

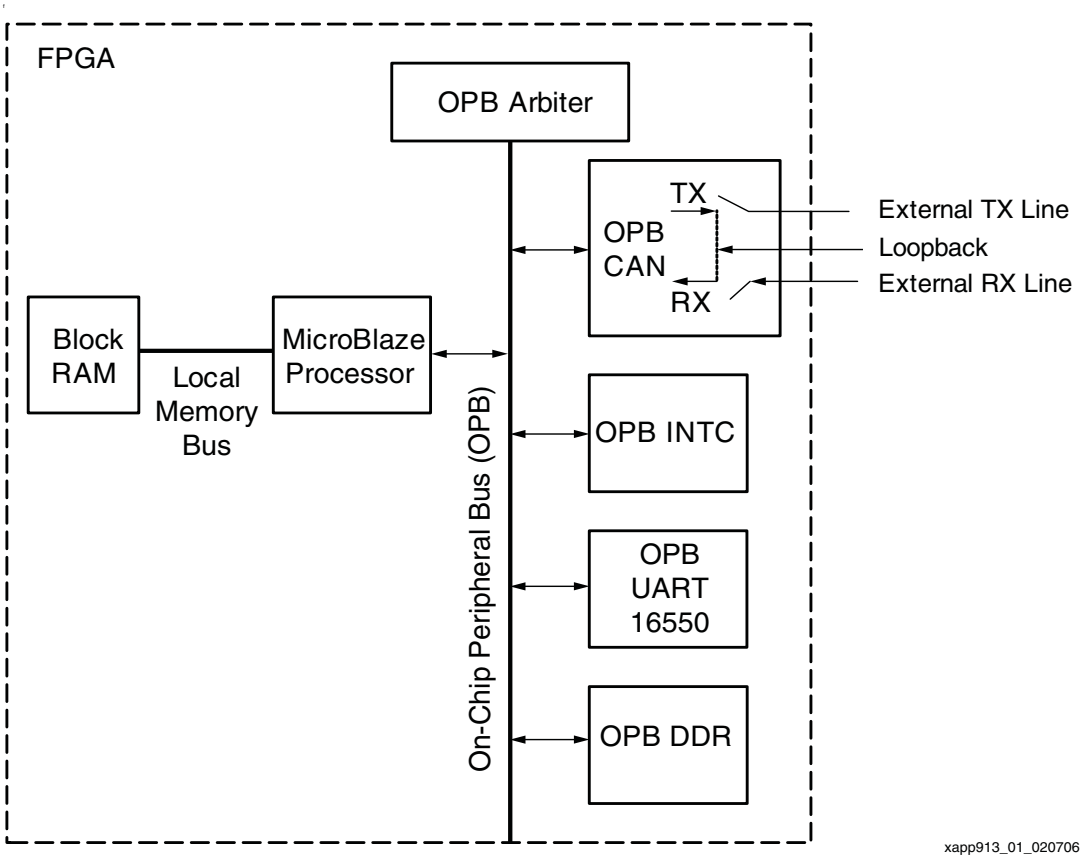

Figure 1: **Reference System Block Diagram**

Address Map

Table 1: **Reference System Address Map**

Peripheral	Instance	Base Address	High Address
opb_mdm	debug_module	0x41400000	0x4140FFFF
lmb_bram_if_cntlr	dlmb_cntlr	0x00000000	0x00007FFF
lmb_bram_if_cntlr	ilmb_cntlr	0x00000000	0x00007FFF
opb_ddr	ddr_sdram_64Mx32	0x28000000	0x280FFFFF
opb_uart16550	RS232_UART	0x40400000	0x4040FFFF
opb_can	OPB_CAN_0	0x80010000	0x800100FF
opb_intc	OPB_INTC_0	0x80020000	0x8002FFFF

Setting up a system with the OPB CAN Controller

The OPB CAN is configured for a FIFO depth of 2 and uses a single acceptance filter. The system uses a crystal oscillator to externally generate the CAN_CLK signal. The DDR SDRAM is used to store the software application. It operates at a frequency different from the OPB CLK frequency. The interrupt from the OPB CAN core is connected to the interrupt line of the OPB INTC. The OPB UART16550 is used for the display of messages on the Hyperterminal. This section describes how to set up the EDK system with these cores.

Ports and Parameters for the Cores

All the cores of the system are slaves on the OPB bus. The parameter C_BASEADDR of the OPB CAN controller is `0x80010000` and the parameter C_HIGHADDR is `0x800100FF`. The IP2Bus_IntrEvent port of the OPB CAN is connected to the Intr port of the interrupt controller (CAN_Interrupt signal). The interrupt request port of the interrupt controller, Irq, is connected to the interrupt port of the MicroBlaze (interrupt_mb signal).

Figure 2 shows the parameter settings for the OPB CAN controller. These parameters correspond to the smallest FIFO depth and the minimum number of acceptance filters for the CAN core. The RX and TX FIFO depths should always be equal and have a value of 2^n, where n is between 1 and 6. Therefore, the maximum configurable FIFO depth is 64. Note that the parameters for FIFO depth cannot be set to 0. There are 4 acceptance filters for the OPB CAN core, hence the parameter for the number of acceptance filters is assigned a value between 0 (no acceptance filters used) and 4.

.

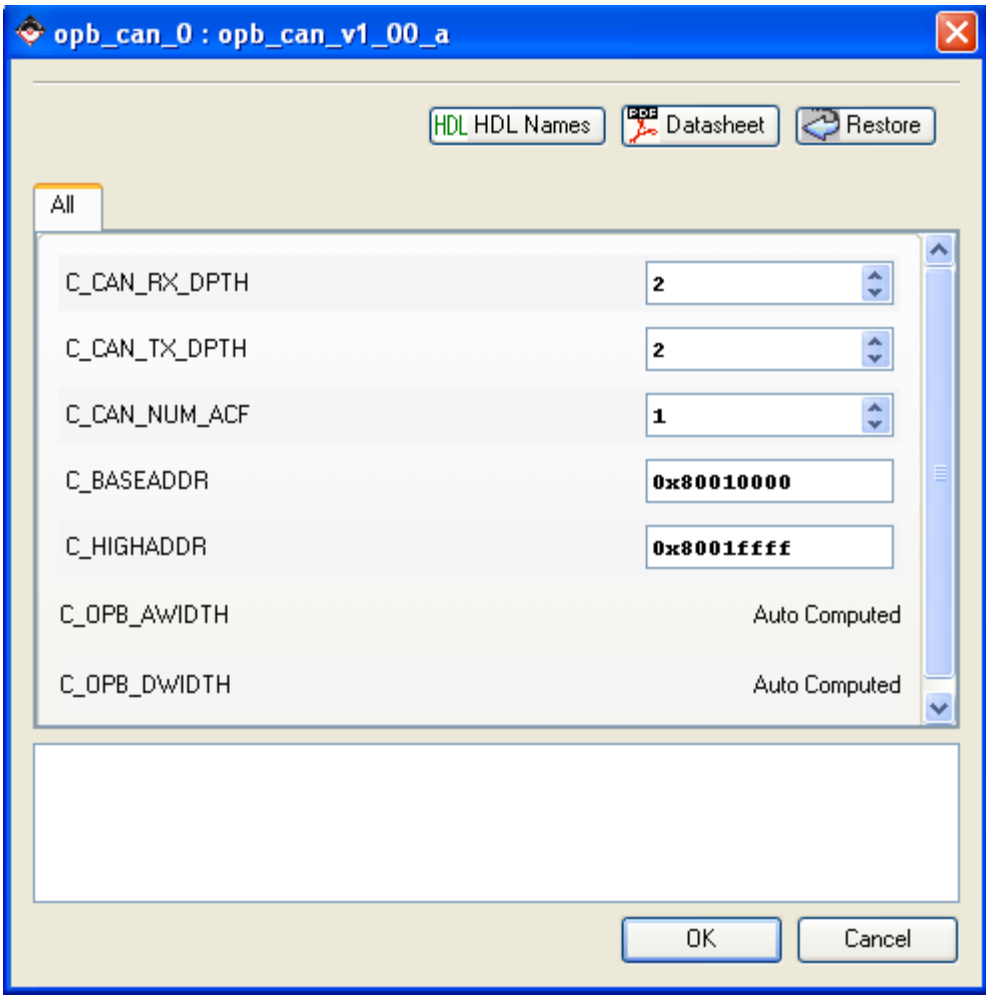

Figure 2: **Parameter Settings for OPB CAN**

Clocking for Reference System

The SP305 board provides a 100 MHz clock. The 100 MHz clock is fed into a DCM, which creates the 50 MHz clock. The 50 MHz clock, which is the system clock, is fed into the processor and the other cores in the system.

In this reference design, an external crystal is used to generate the 24 MHz CAN clock which is fed to the CAN_CLK pin of the OPB CAN (signal opb_can_0_CAN_CLK). The CAN CLK can have a frequency between 8MHz and 24MHz.

The Software Application

The software application provided with this reference system works in the interrupt mode. The application calls the OPB CAN driver functions in order to set the CAN controller and the interrupt system for a loopback application. To begin, the OPB CAN controller is initialized with the device ID. Following initialization, the CAN core is set to the Configuration mode and the Bit Timing Register (BTR) and Baud Rate Prescaler Register (BRPR) registers are written to in order to set the Baud Rate to 500Kbps. Then the interrupt handlers are set, the interrupt system is set up, and all the interrupts of the OPB CAN core are enabled. The OPB CAN core is then set to Loopback mode. The Acceptance Filter Register (ACR) for the parameter C_CAN_NUM_ACF = 1 is written. The Acceptance Filter ID Register (AFIR) has the same identifier as the transmitted frame. Hence, the frame transmitted should be received by the Receiver. The Acceptance Filter Mask Register (AFMR) has all 1's, so that all bits of the transmitted identifier are compared to the bits of the AFIR. A flag called LoopbackError, if TRUE, indicates that Loopback was a failure.

To begin loopback, a frame is transmitted which results in the calling of the Transmit Interrupt Handler (SendHandler).The SendDone flag is set to a TRUE value once the SendHandler is called. The flag LoopbackError is set to TRUE if Transmission is not a success.

Following transmission in loopback mode, the RXNEMP interrupt is set which calls the Receive Interrupt Handler (RecvHandler).The RecvDone flag is set to a TRUE value within the RecvHandler.The flag LoopbackError is set to TRUE if Reception is not a success or the frames received do not match with the frames transmitted.

The application waits until both Transmission and Reception are completed (SendHandler and RecvHandler both are set to TRUE) and then checks if the LoopbackError flag has been set to TRUE. If so, it exits the programs with the message that the Loopback was a failure, else it exits with a message that Loopback was a success.

The application prints status messages at different points in the program through the OPB UART16550 core.

Figure 3 illustrates the implementation flow of the software application for the OPB CAN operating in loopback.

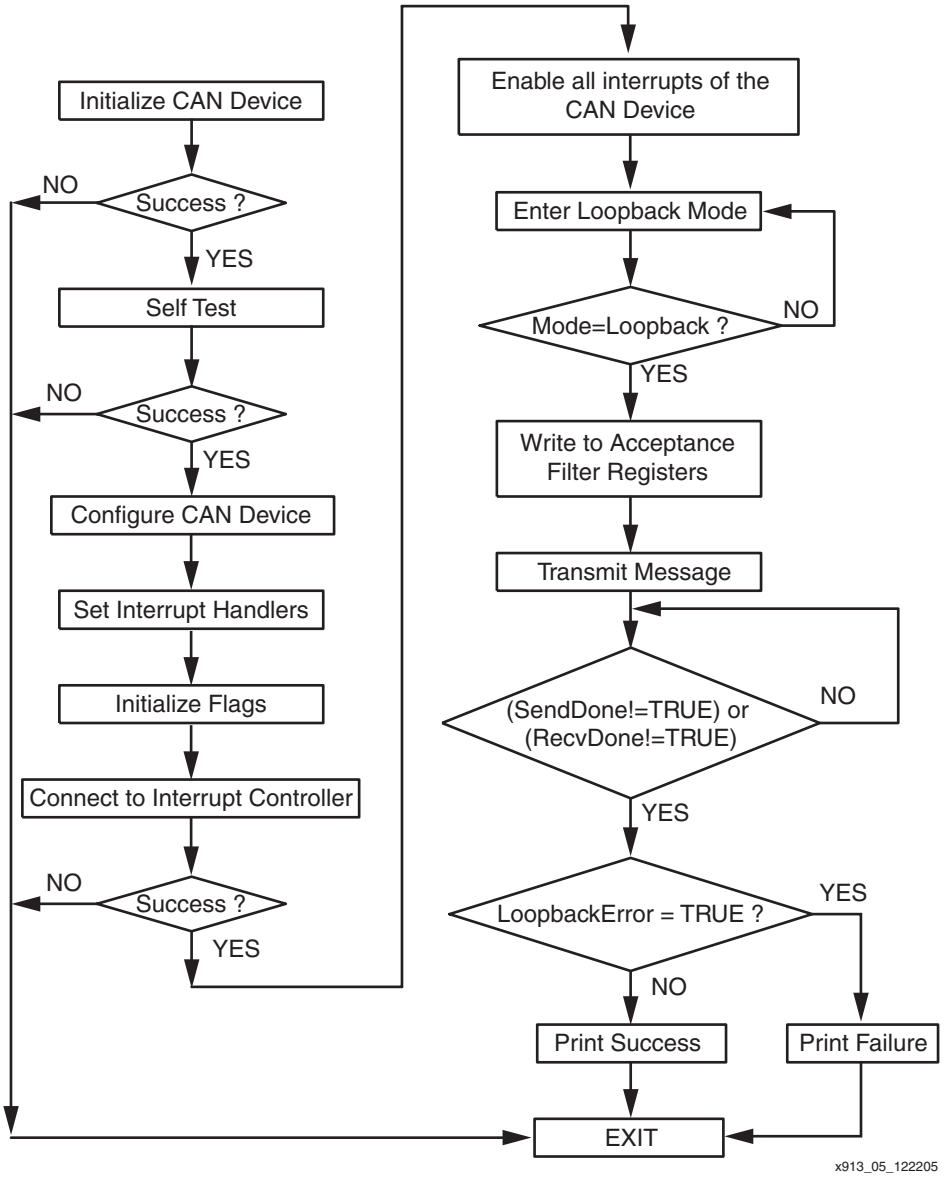

Figure 3: **Software Implementation Block Diagram**

Executing the Reference System

To execute the reference system, the SP305 board must be set up correctly, the bitstream must be generated, and the software application must be compiled. A pre-built bitstream, `download.bit`, and the compiled software application, `executable.elf`, are available in `ready_for_download` under the project root directory. Using HyperTerminal or a similar serial communications utility, map the utility's operation to the physical COM port to be used, then connect the board's UART port to this COM port. The terminal settings must have the baud rate set to 9600, data bits to 8, parity to None, and flow control to None. See Figure 4 for the HyperTerminal settings.

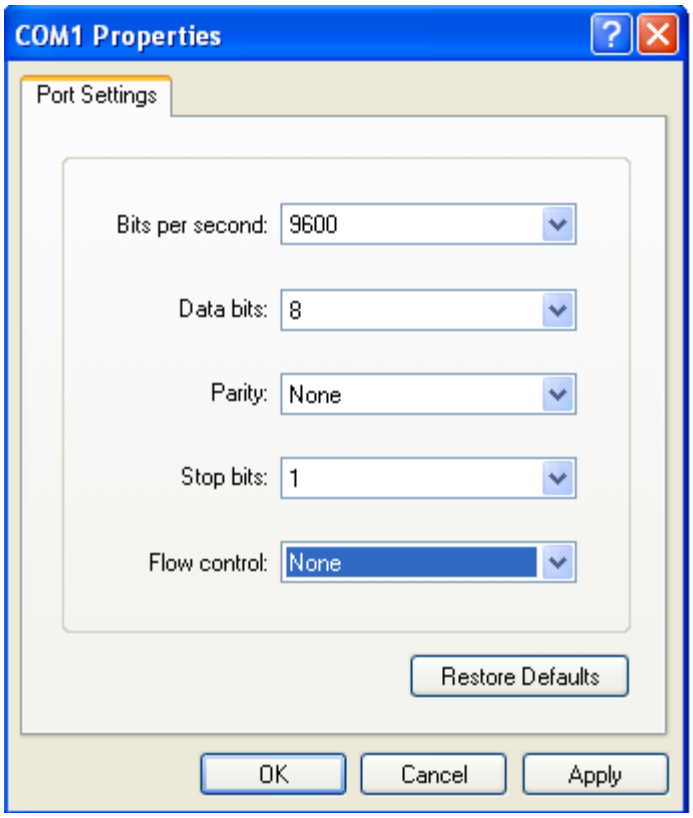

Figure 4: **Hyperterminal Settings**

After downloading the hardware bitstream, establish an XMD connection. Once connected, the compiled software application, executable.elf, must be downloaded and executed. The status of the software application is displayed in the HyperTerminal data screen. At the end of the software application, the output at the Hyperterminal should read the following:

```
Entered main
OPB CAN Initialized
OPB CAN Controller Configured
All interrupts enabled
OPB CAN set in Loopback Mode
Acceptance Filter Registers Written
Frame Transmitted in Loopback Mode
Loopback was a Success
Exited main
```

References

1. *OPB CAN* (v1.00.a) Design Specification, DS523

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
02/10/06	1.0	Initial Xilinx release.