XAPP924 (v1.2) June 5, 2007

Reference System: Using the OPB EPC with the SMSC LAN 91C111 Controller

Author: Sundararajan Ananthakrishnan

Abstract

This application note demonstrates the use of On-Chip Peripheral Bus (OPB) External Peripheral Controller (EPC) to support the SMSC LAN 91C111 controller chip in a PowerPC® 405 processor based reference system.

This application note describes the following:

- OPB EPC parameter settings for the SMSC LAN 91C111 controller
- Interrupt control logic that handles the SMSC LAN 91C111 controller's interrupt signal
- External glue logic required by the OPB EPC core for interfacing with the SMSC LAN 91C111 controller
- Standalone software application for testing the SMSC LAN 91C111 controller
- Porting of SMSC LAN 91C111 VxWorks driver for the PowerPC 405 processor based reference system

Included Systems

Included with this application note is the reference system for the Memec Design Virtex™-II Pro P7-ff672 Development Board Rev 2 with P160 Comm module. The reference system is available at:

- www.xilinx.com/bvdocs/appnotes/xapp924.zip

Introduction

The OPB EPC is used to control peripherals that are connected externally to Xilinx FPGAs. The most commonly used external devices are LAN controllers, USB controllers and IEEE 1394 (FireWire) controllers.

The OPB EPC is a OPB slave only device and does not support any DMA operations from external devices. The OPB EPC supports both multiplexed and non-multiplexed address/data buses, where the data bus width can be 8,16, or 32 bits wide.

The standalone software application provided with this reference system is executed from the cacheable region of the external memory. The reference system uses PLB SDRAM controller to control the external SDRAM memory.

The SMSC LAN 91C111 controller used in the reference systems is a 10/100 Non-PCI Ethernet single LAN controller. The SMSC LAN 91C111 controller interfaces with the OPB EPC in the asynchronous mode of operation.

Hardware and Software Requirements

The hardware and software requirements are:

- Memec Design Virtex-II Pro P7-ff672 Development Board Rev 2 with P160 Comm module.
- Xilinx Platform USB cable or Parallel IV programming cable
- RS232 serial cable and serial communication utility (HyperTerminal)
- Xilinx Platform Studio 9.1.01i
- Xilinx Integrated Software Environment (ISE™) 9.1.03i

Reference System Specifics

The Memec Design Virtex-II Pro P7-ff672 Development Board Rev 2 based reference system has a PowerPC 405 processor with caches enabled. The PLB2OPB Bridge is used to connect the OPB EPC to the PowerPC 405 processor. The OPB UART Lite core, the OPB Interrupt Controller(OPB INTC), and the OPB Timer cores are also used in the reference system.

The reference system is shown in Figure 1 and the address map of the reference system is shown in Table 1.

Block Diagram

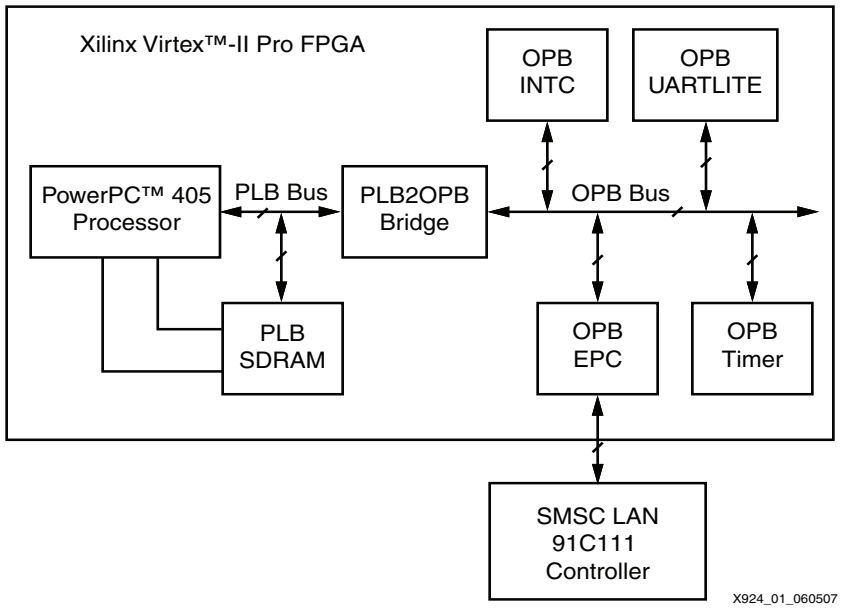

Figure 1: **Reference System Block Diagram**

Address Map

Table 1: **Reference System Address Map**

Peripheral	Instance	Base Address	High Address
plb_sdram	SDRAM_8Mx32	0x00000000	0x01FFFFFF
plb_bram_if_cntlr	plb_bram_if_cntlr_1	0xFFFF0000	0xFFFFFFFF
opb_epc	opb_epc_0	0x8001C000	0x8001FFFF
opb_uartlite	RS232	0x8000FF00	0x8000FFFF
opb_intc	opb_intc_0	0x41200000	0x4120FFFF

System Configuration

This Memec Design Virtex-II Pro P7-ff672 Development Board Rev 2 based reference system has the PowerPC 405 processor with the caches enabled. The PLB2OPB bridge connects the OPB EPC to the PowerPC 405 processor in the reference system. The PLB SDRAM is used as the external memory and is configured to allow cacheline transactions from the PowerPC 405 processor in the reference system.

Setting the OPB EPC Common Parameters

The OPB EPC can support up to a maximum of four peripherals. In the reference system, since the SMSC LAN 91C111 controller is the only external peripheral connected to the OPB EPC, the parameter *Number of External Peripherals,* is set to 1.

Since the address bus of the SMSC LAN 91C111 controller connected to the OPB EPC system is four bits wide, the parameter *Maximum Address Bus Width of All External Peripherals* is set to 4.

Because the SMSC LAN 91C111 controller has an 8-bit data bus, the parameter *Maximum Data Bus Width of All External Peripherals,* is set to 8. The SMSC LAN 91C111 controller is the only external peripheral that is connected in this reference system, therefore, the parameter *Maximum Data Bus Width and Address/Data-Multiplexed Address Bus Width of All External Peripherals* is also set to 8.

Since the OPB EPC does not support burst transactions from or to the external devices, the parameter *Enable Burst Support in External Peripherals* is not enabled.

Figure 2 shows the parameter settings for the OPB EPC when the **User Common** tab is selected.

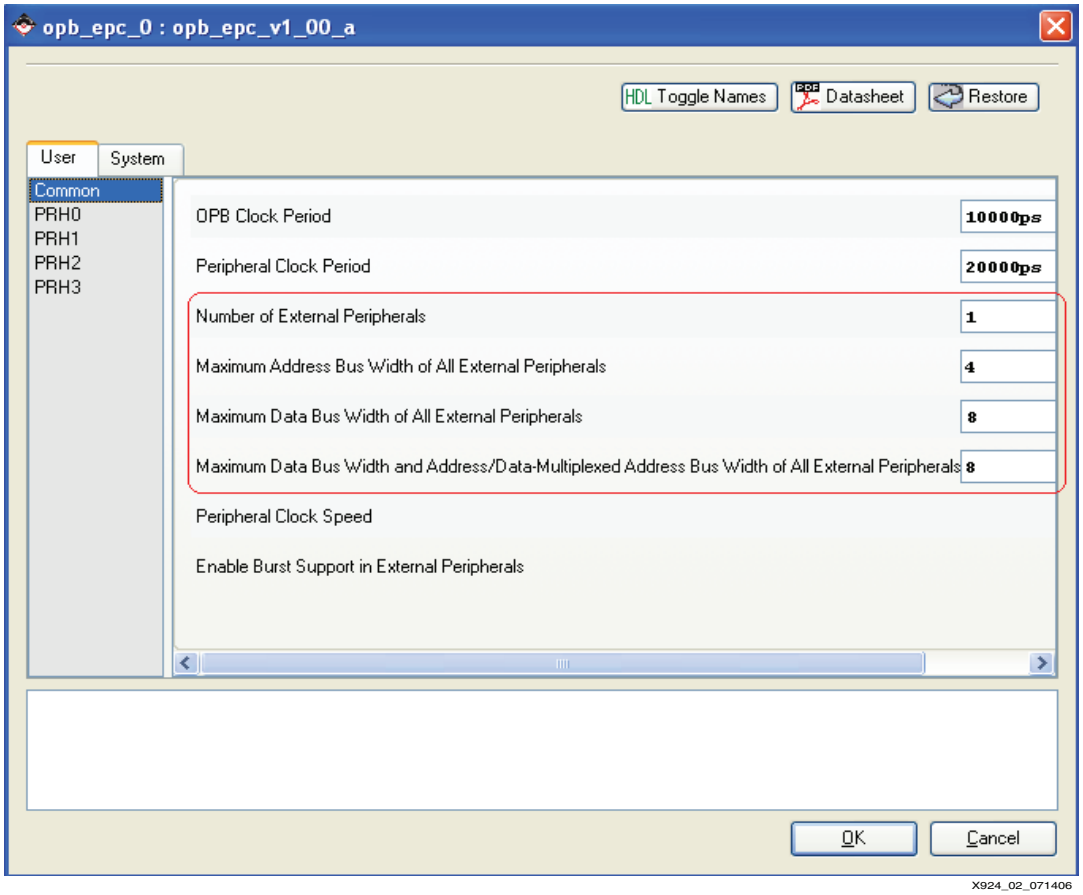

Figure 2: **Setting the OPB EPC Common Parameters**

Setting the OPB EPC Peripheral (PRH0) Parameters

Because the SMSC LAN 91C111 controller is the only external peripheral connected to the OPB EPC, the parameter settings are done only for the Peripheral 0 (PRH0).

The OPB EPC does not support access to the external FIFO within the SMSC LAN 91C111 controller, therefore, the parameter, *Support FIFO Access in External Peripheral*, is not enabled, and the parameter, *FIFO Offset from Base Address,* is set to 0.

Since the SMSC LAN 91C111 controller has a 4-bit address bus, the parameter *Address Bus Width of External Peripheral* is set to 4.

The data bus width of the SMSC LAN controller in the P160 comm module is 16. However only 8 data lines are routed in the Memec Virtex-II Pro P7-ff672 Development Board to the FPGA. Hence the data width parameter in the OPB EPC *Data Bus Width of External Peripheral* is set to 8. To match the 16-bit data interface of the SMSC LAN91C111 controller, the P160 Comm Module replicates the lower order data byte0 (i.e. bits 0-7) from the FPGA onto the higher order data byte byte1 (i.e. bits 8-15) of the SMSC LAN 91C111 controller.

The data bus width of SMSC LAN 91C111 controller is 8-bits wide and the OPB data bus is 32-bits wide. The parameter *Support Multiple Cycle Access in Peripheral to Match an OPB Cycle* is enabled thereby enabling the OPB EPC to run multiple cycles on the peripheral interface for a single OPB read/write cycle.

The OPB EPC can operate in both synchronous and asynchronous modes. In the reference system, the OPB EPC is configured to asynchronous mode of operation by not enabling the parameter *Peripheral Access Mode*.

The OPB EPC core supports both multiplexed and non-multiplexed address and data buses, where the data bus width can be 8, 16 or 32 bits. Since the SMSC LAN 91C111 controller has a separate address and data bus, the parameter *Peripheral Bus Type* is not enabled.

The timing parameters for the OPB EPC are set for the asynchronous mode of operation. The timing values are set by referring to the datasheet of the SMSC LAN 91C111 controller.

Figure 3 shows the OPB EPC peripheral interface parameter settings in the **User PRH0** tab.

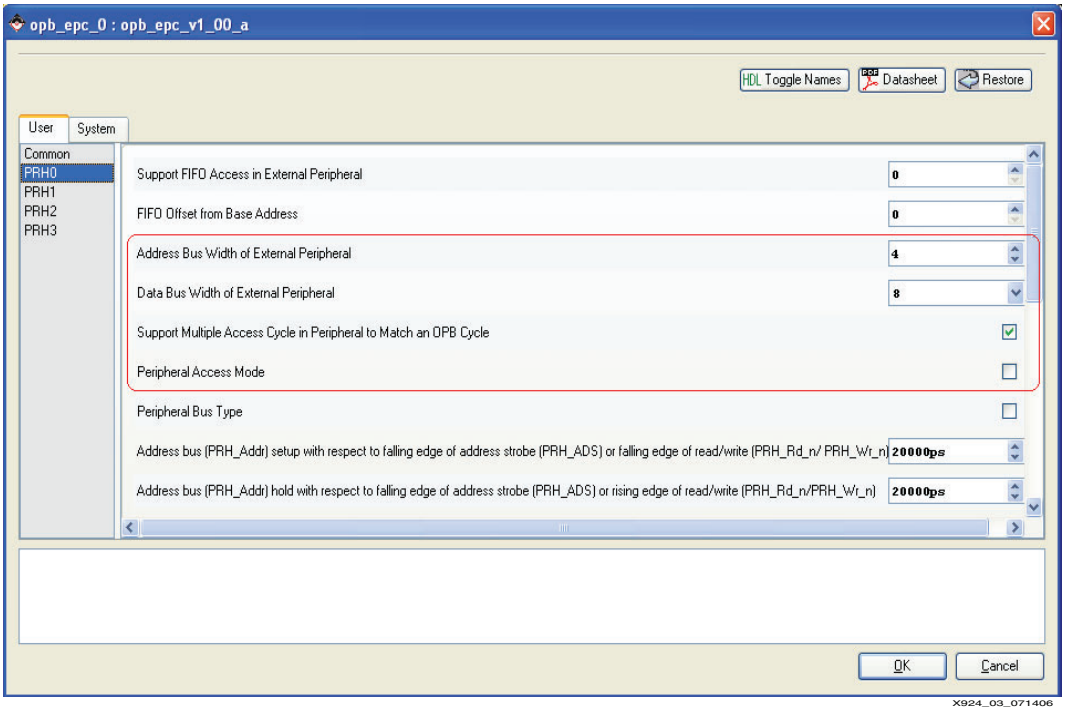

Figure 3: **Setting the OPB EPC Peripheral (PRH0) Parameters**

Interrupt Control Logic

The SMSC LAN 91C111 controller generates an interrupt signal depending on several conditions.The OPB EPC does not support routing of the interrupt through the OPB EPC core, therefore, the interrupt signal is directly routed to the processor through the OPB INTC core in the system. The application software running on the PowerPC 405 processor manages the interrupt handling.

Obtain the list of interrupts by expanding the opb_intc_0 tree node and clicking on the last port under the **Net** for **Intr**. This will bring up the **Interrupt Connection Dialog** box. Add the **ext_int_p160_SMSC** external interrupt input signal to Connected Interrupts of the OPB INTC as shown in Figure 4

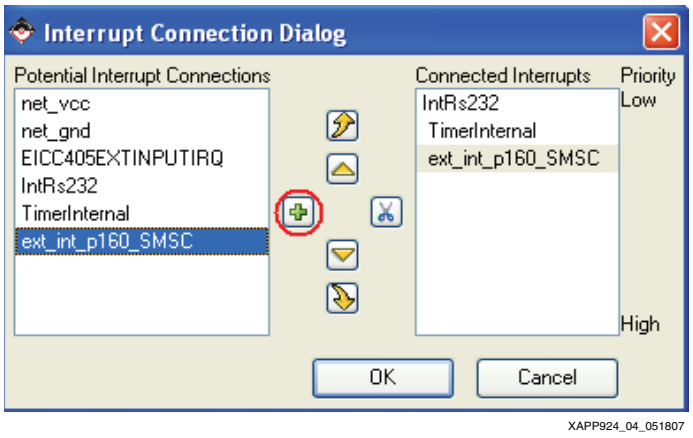

Figure 4: **SMSC LAN 91C111 Controller Interrupt Signal Connection**

External glue logic

The OPB EPC `PRH_BE` output signals are active high while the SMSC LAN91C111 controller byte enable input signals are active low. To match this interface requirement, additional logic is required to invert the `PRH_BE` signals before they are connected to the byte enable signals of SMSC LAN91C111 controller. This is done by using the Utililty logic IP cores available in the EDK environment (util_vector_logic_v1_00_a).

Software Application

This section describes the software application that is used for testing the SMSC LAN 91C111 Controller in this reference system. The software application is found under the project root directory, *Test_App/smsc_emac91_examples.c.*

The software application is executed from the external memory and Mark to Initialize BRAM's must be disabled. The linker script has all the options set to load the external memory.

The software application used for testing the SMSC LAN91C111 controller is based on the *Functional tests and Diagnostics* that are described in the SMSC LAN91C111 Ethernet Controller Technical Reference Manual.

The tests contained in the following section tests the major blocks of the MAC and the PHY block of the SMSC LAN 91C111 controller.

Emac91_MACRegisterTest

This routine writes and reads from the Write/Read bits of the MAC registers and ignores all the Reserved/Read-Only bits

Emac91_RamBufferTest

This routine tests the Packet Memory of the SMSC LAN91C111 controller.

Emac91_LoopBackTest

This routine tests the SMSC LAN91C111 controller in all loopback modes.

The various loopback modes tested are:

- *EPH Loopback Test*: This is the internal loopback at the MAC level.
- *PHY Loopback Test*: This is the diagnostic loopback at the PHY level.

Emac91_MMUTest

The Emac91_MMUTest has multiple routines that tests various MMU commands:

- *Emac91_MMUAllocateTxMemory*: This tests the MMU Command *Allocate Memory for Tx*.
- *Emac91_MMUReset*: This tests the MMU Command *Reset MMU*
- *Emac91_MMURemoveFrameRxFIFO*: This routine tests the MMU Command *Remove the Frame from top of the Rx FIFO*.
- *Emac91_MMURemoveReleaseRxFIFO*: This routine tests the MMU Command *Remove and release top of the Rx FIFO*.
- *Emac91_MMUReleasespecificpacket*: This routine tests the MMU Command *Release specific packet* command and checks for the default values of the FIFO port registers and Memory Information register.
- *Emac91_MMUTxpacketEnqueue*: This routine tests the MMU command *Enqueue Packet into the TX FIFO*
- *Emac91_MMUResetTxFIFO*: This routine tests the MMU command *Reset the Tx FIFO*

Emac91_PHYTest

This routine writes and reads from the Write/Read bits of some of the internal PHY registers and ignores all the Reserved/Read-Only bits

Note: There are some changes in the procedure in some of the tests as compared to the procedure in the *SMSC LAN 91C111 Technical Reference Manual*. All these deviations are noted down in the notes section of the C source files. The SMSC LAN 91C111 Technical Reference Manual can be found on the SMSC website.

Executing the Reference System

The Memec Virtex-II Pro PowerPC 405 reference system has two EDK project files. The *system.xmp* is the EDK project file for the stand-alone application and the *system_vxworks.xmp* is the EDK project file for the VxWorks application.

This section outlines the stand-alone application that is a part of the system.xmp project.

To execute the reference systems, the bitstreams need to be generated and the software applications compiled. The bitstreams and the compiled software applications for these systems are available in `ready_for_download` directory under each project root directory.

A HyperTerminal or similar program needs to be connected to the COM port and the board's RS232 needs to be connected to the COM port. Set the HyperTerminal to the Baud Rate of **9600**, Data Bits to **8**, Parity to **None** and Flow Control to **None** as shown in Figure 5.

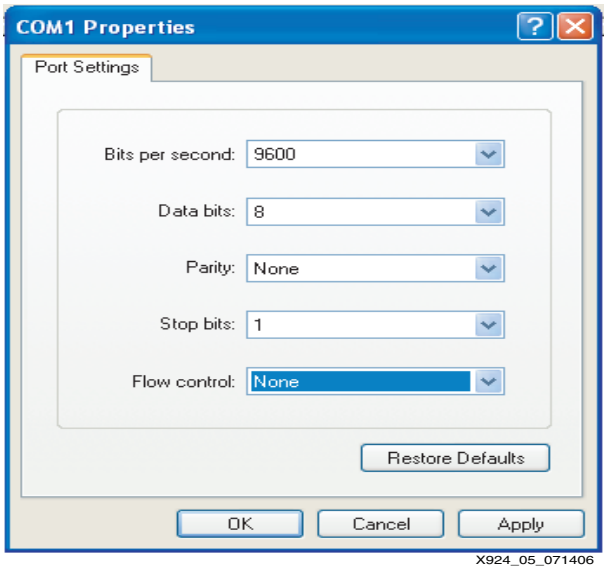

Figure 5: **HyperTerminal Settings**

Executing the Reference System using the Pre-Built Bitstream and the Compiled Software Applications

To execute the system using files inside the `ready_for_download/` in the project root directory, follow these steps:

1. Change directories to the `ready_for_download` directory.
2. Use iMPACT to download the bitstream by using the following:

   ```
   impact -batch xapp924.cmd
   ```

3. Invoke XMD and connect to the PowerPC 405 processor by the following command:

   ```
   xmd -opt xapp924.opt
   ```

4. Download the executables by the following command:

   ```
   dow executable.elf
   ```

Executing the Reference System from EDK

To execute the system using EDK, follow these steps:

1. Open `system.xmp` inside EDK.
2. Use **Hardware→Generate Bitstream** to generate a bitstream for the system.
3. Use **Software→Build All User Applications** to build the software applications.
4. Download the bitstream to the board with **Device Configuration→Download Bitstream**.
5. Launch XMD with **Debug→Launch XMD...**
6. Download the executables by the following command:

   ```
   dow executable.elf
   ```

Running the Software Applications

To run the either of software applications, use the ***run*** command inside XMD. The status of the software application is displayed in the HyperTerminal data screen.

```
The MAC Register Test has completed Successfully
The RAM Buffer Test has completed Successfully
```

```
The EPH Loopback Test has completed Successfully
The PHY Loopback Test has completed Successfully
The MMU Tests have completed Successfully
The PHY Tests have completed Successfully

The SMSC LAN Test Examples have passed Successfully
```

VxWorks Driver for the SMSC LAN91C111 controller

This section outlines the steps to be taken for using the the VxWorks driver for SMSC LAN91C111 controller in the Xilinx OPB EPC PowerPC 405 reference system.

The driver has been ported to the Xilinx OPB EPC PowerPC 405 reference system from the existing VxWorks driver provided by SMSC for an Intel platform. The source code for the VxWorks driver (SRC-111-VX Source Code) is available as a zip file (*111-vxworks.zip*) on the SMSC website.

Note: The pre-compiled bootable VxWorks image for this reference system is available under the Project root directory, vxworks/`vxWorks`. This image can be downloaded directly to the reference system after downloading the bitstream.

Creating the Tornado 2.x (VxWorks 5.x) BSP

1. Open the system_vxworks project in EDK. The system_vxworks project contains all the settings required for generating the VxWorks BSP for the reference system. This project directory will hereby be referred as <edk_proj>.
2. Clean the SW libraries with **Software--->Clean Libraries.**
3. Generate the VxWorks BSP with **Software--->Generate Libraries and BSPs.**

   The resulting VxWorks BSP is located in **<edk_proj>/pc405_0/bsp_ppc405_0.**

   This directory will hereby be referred as <vxworks_bsp>.
4. Exit EDK.

Further information about creating a Tornado 2.x (VxWorks 5.x) Board Support Packages is given in *OS and Libraries Document Collection (oslibs_rm.pdf)*, which is avaliable as a part of the EDK installation.

Integrating the SMSC VxWorks driver into the VxWorks BSP

This zip file(*111-vxworks.zip*) contains the following files.

- lan91c111End.c - SMSC LAN91C111 END source file for an Intel platform
- lan91c111End.h - SMSC LAN91C111 END header file for an Intel platform
- Makefile - SMSC make file for building this driver
- readme.pdf - A file consisting of the changes to be made to integrate this driver to the VxWorks BSP

The changes suggested in the readme.pdf for integrating the driver in the VxWorks BSP tailored for this OPB EPC reference system are:

1. Copy the lan91c111End.c and lan91c111End.h files to the BSP directory <vxworks_bsp>.
2. Add "lan91c111End.o" to the MACH_EXTRA line in the file "/<vxworks_bsp>/Makefile"
   - Change

   ```
   MACH_EXTRA    += $(patsubst %.c, %.o, $(wildcard ppc405_0_drv_*.c))
   ```

   to

   ```
   MACH_EXTRA    += $(patsubst %.c, %.o, $(wildcard ppc405_0_drv_*.c))
   MACH_EXTRA    += lan91c111End.o
   ```
3. Add "#define INCLUDE_LAN91C111_END" in the file "/<vxworks_bsp>/config.h"

4. Change the default bootline parameters in the file "/<vxworks_bsp>/config.h"

   Change

   ```
   #define DEFAULT_BOOT_LINE \
   "todo(0,0)host:vxWorks h=192.168.0.1 e=192.168.0.2 u=xemhost"
   ```

   to

   ```
   #define DEFAULT_BOOT_LINE \
   "lnc(0,0)host:vxWorks h=192.168.0.1 e=192.168.0.2 u=xemhost"
   ```

   Where:

   - lnc is name of the SMSC LAN 91C111 controller.
   - 192.168.0.1 is the IP address of the Host. The IP address of the Host machine needs to be specified here.
   - 192.168.0.2 is the IP address of the Target (The reference system in which the SMSC LAN 91C111 controller is present). The Target IP address needs to be changed based on the local network configuration.

5. Change the console baud rate in "/<vxworks_bsp>/config.h"

   Change

   ```
   #define CONSOLE_BAUD_RATE 19200
   ```

   to

   ```
   #define CONSOLE_BAUD_RATE 9600
   ```

6. Add the following to the file /<vxworks_bsp>/configNet.h before the endDevTbl

   ```
   #ifdef INCLUDE_LAN91C111_END
   #define END_LAN91C111_LOAD_FUNC lan91c111Load
   #define END_LAN91C111_BUFF_LOAN 1
   /*<unit>:<IO>:<intVec>:<intLevel>:<offset>:<configValue>*/
   #define END_LAN91C111_LOAD_STRING
   "0x8001C000:0x0:0xB:0x0:0x3100:00:12:79:D1:C1:41"
   IMPORT END_OBJ * END_LAN91C111_LOAD_FUNC (char *, void*);
   #endif /* INCLUDE_LAN91C111_END */
   ```

   The definition of parameters passed to the END_LAN91C111_LOAD_STRING are as follows:

   - "0x8001C000" is IOBase and is the base address for the SMSC LAN 91C111 controller. This needs to be set to the base address of the SMSC LAN 91C111 controller in the reference system.
   - "0x0" is the interrupt vector number of the SMSC interrupt. This needs to be set based on the way the SMSC interrupt is connected in the hardware.
   - "0xB" is the interrupt level and is a don't care in this system.
   - "0x0" is the offset in a cluster and is a don't care in this system.
   - "0x3100" is the config value. This is the initial configuration value for the SMSC LAN 91C111 controller for Transmit and Receive settings. This particular configuration value enables Auto Negotiation and Disables Early Transmit / Receive.
   - "00:12:79:D1:C1:41" is a user defined MAC address. This has to be a valid MAC address. If a config EEPROM is connected to the SMSC LAN 91C111 controller then the MAC address SHOULD NOT be specified in the END_LAN91C111_LOAD_STRING.

7. Add the following to endDevTbl in the file "/<vxworks_bsp>/configNet.h"

   ```
   #ifdef INCLUDE_LAN91C111_END
   {0, END_LAN91C111_LOAD_FUNC, END_LAN91C111_LOAD_STRING,
   END_LAN91C111_BUFF_LOAN, NULL, FALSE},
   #endif /* INCLUDE_LAN91C111_END */
   ```

Porting the VxWorks Driver to the PowerPC 405 reference system

There are changes required to be made to the driver source file (lan91c111End.c) for the following reasons:

- The VxWorks END driver for SMSC LAN 91C111 controller provided by SMSC is for an Intel x86 architecture
- The Xilinx reference system is based on PowerPC 405 architecture.
- The VxWorks END driver for SMSC LAN 91C111 controller provided by SMSC is for an ISA bus based system.
- Limitations in the hardware system.
- Remove compiler warnings

1. Include the xio.h header file.

   ```
   #include xio.h
   ```

2. Include the xparameters.h header file.

   ```
   #include xparameters.h
   ```

3. Add the functions lan91c111InWord and lan91c111OutWord

   These two functions swap the bytes while reading/writing the data from/to the SMSC LAN 91C111 controller.

   ```
   Xuint16 lan91c111InWord(Xuint32 BaseAddress)
   {
       Xuint16 Temp;
       Temp = XIo_In16(BaseAddress);
       return (((((Temp) << 8) & 0xFF00) | (((Temp) >> 8) & 0x00FF)));
   }

   void lan91c111OutWord(Xuint32 BaseAddress, Xuint16 Data)
   {
       Xuint16 Temp;
       Temp = (((((Data) << 8) & 0xFF00) | (((Data) >> 8) & 0x00FF)));
        XIo_Out16(BaseAddress, Temp);
   }
   ```

   - Change

   ```
   #define SYS_OUT_SHORT(addr,value) sysOutWord(addr, value);
   ```

   to

   ```
   #define SYS_OUT_SHORT(addr,value) lan91c111OutWord(addr, value);
   ```

   - Change

   ```
   #define SYS_IN_SHORT(port, value_addr) (*value_addr = sysInWord(port))
   ```

   to

   ```
   #define SYS_IN_SHORT(port, value_addr) (*value_addr =
   lan91c111InWord(port))
   ```

   Note: These changes are to offset the byte swapping done in the hardware. The interface of the OPB EPC to the SMSC LAN91C111 controller in this reference system results in this byte swapping. If there is NO byte swapping in the hardware then there is no requirement of making these changes.

4. There are few instances where the data written to the device SHOULD NOT be byte swapped and hence the macros (SYS_OUT_SHORT and SYS_IN_SHORT) CANNOT be used. The functions sysOutWord and sysInWord are used instead.

   - Change (In the function lan91c111WriteToChip)

```
if(length > 1)
{
     SYS_OUT_SHORT((DataPort),(*((PUSHORT) p)));
     length -= sizeof(USHORT);
     p += sizeof(USHORT);
}
```

   to

```
if(length > 1)
{
      sysOutWord((DataPort),(*((PUSHORT) p)));
      length -= sizeof(USHORT);
      p += sizeof(USHORT);
}
```

   - Change (In the function lan91c111WriteToChip)

```
SYS_OUT_SHORT((DataPort),((USHORT)CTL_BYTE_CRC<<8));
```

   to

```
sysOutWord((DataPort),((USHORT)CTL_BYTE_CRC<<8));
```

   - Change (In the function AdapterReset)

```
SYS_OUT_SHORT((IOBase + BANK1_IA0), *EthAddress++);
SYS_OUT_SHORT((IOBase + BANK1_IA2), *EthAddress++);
SYS_OUT_SHORT((IOBase + BANK1_IA4), *EthAddress);
```

   to

```
sysOutWord((IOBase + BANK1_IA0), *EthAddress++);
sysOutWord((IOBase + BANK1_IA2), *EthAddress++);
sysOutWord((IOBase + BANK1_IA4), *EthAddress++);
```

   - Change(In the function AdapterReset)

```
SYS_IN_SHORT(IOBase + BANK1_IA0, EthAddress++);
SYS_IN_SHORT(IOBase + BANK1_IA2, EthAddress++);
SYS_IN_SHORT(IOBase + BANK1_IA4, EthAddress);
```

   to

```
*EthAddress++ = sysInWord (IOBase + BANK1_IA0);
*EthAddress++ = sysInWord (IOBase + BANK1_IA2);
*EthAddress = sysInWord (IOBase + BANK1_IA4);
```

   Note: These changes are to offset the byte swapping done in the hardware. The interface of the OPB EPC to the SMSC LAN91C111 controller in this reference system results in this byte swapping. If there is NO byte swapping in the hardware then there is no requirement of making these changes.

5. The asm instruction(cli) for clearing the interrupts in x86 architecture should be commented out from the function EarlyTransmit

   - Change

```
__asm("cli");
```

   to

```
/* __asm("cli"); */
```

6. Redefine sysDelay, which is a delay in x86 architecture, to lan91c111Delay:

```
#define sysDelay lan91c111Delay
```

7. Comment out the delay inside the function lan91c111Delay.
8. The function for enabling the interrupts in a x86 architecture should be changed:
   - Change

   ```
   #define SYS_INT_ENABLE(pDrvCtrl) {sysIntEnablePIC(pDrvCtrl->ilevel);}
   ```

   to

   ```
   #define SYS_INT_ENABLE(pDrvCtrl) {intEnable(pDrvCtrl->ivec);}
   ```
9. Add LOCAL to the function definition of the functions lan91c111Recv and lan91c111Ioctl to remove compiler warnings.
   - Change

   ```
   STATUS lan91c111Recv
   ```

   to

   ```
   LOCAL STATUS lan91c111Recv
   ```
   - Change

   ```
   int lan91c111Ioctl
   ```

   to

   ```
   LOCAL int lan91c111Ioctl
   ```
10. The SMSC LAN driver is for an x86 architecture ISA based card and hence the base addresses are USHORT. This has to be changed to UINT so that 32-bit base addresses can be used.Change the type of argument for the base addresses, *IOBase*, *IOAddress* to UINT in the functions, `ReadPhyRegister, WritePhyRegister, InputMDO, OutputMDO and DetectPHY`

`Example:`

   - Change

   ```
   UINT ReadPhyRegister(USHORT IOBase, UCHAR phyaddr, UCHAR phyreg);
   ```

   to

   ```
   UINT ReadPhyRegister(UINT IOBase, UCHAR phyaddr, UCHAR phyreg);
   ```

11. The SMSC LAN driver is for an x86 architecture ISA based card and hence the base addresses were USHORT. Remove the typecasting of the Base Address to USHORT in the functions calls to the functions, `ReadPhyRegister, WritePhyRegister, InputMDO, OutputMDO and DetectPHY.`

Example:

   - Change

   ```
   ReadPhyRegister((USHORT)Adapter->IOBase
   ```

   to

   ```
   ReadPhyRegister(Adapter->IOBase
   ```

Compilation

The BSP and the application software for this reference system should be compiled using Tornado 2.x and built without any errors.

Testing the SMSC driver

1. Download the bootable VxWorks image and run the image.
2. Connect the ethernet port of the reference system to the local network.
3. Ping the IP address assigned to the reference system from an external host.

The SMSC driver has been tested with the freely available fping software from the fping site.

The various tested options of `fping` are:

- Pinging for packets of various sizes from 100 bytes to 6000 bytes.
- Pinged with a sweep of 4 to 6000 bytes with 1ms/2 ms time between pings.
- Pinged with random length packets between 4 and 5900 bytes.
- Pinged with 0 wait time for a packet of 1000/2000 bytes.
- Pinged continuously for 12 hours with a sweep between 20 bytes to 5800 bytes.
- Pinged with 5800 bytes and 0 wait time between pings.
- Pinged with 5800 bytes and 0 wait time between pings and 0 wait time for reply
- Pinged with 100 bytes and 0 wait time between pings and 0 wait time for reply

There were a few missing packets in this testing and the percentage of these missing packets varied depending on the packet size, the time between two successive pings, the wait time for reply and the time for which this test was conducted.

Conclusion

This application note describes how to set up the OPB EPC for the PowerPC 405 processor based reference system. The reference system is built for the Memec Design Virtex-II Pro P7-ff672 Development Board Rev 2 with P160 Comm module. The system includes the software application which is used for testing the SMSC LAN 91C111 controller. The application note also illustrates how to use the SMSC LAN91C111 VxWorks driver for the PowerPC 405 processor based OPB EPC reference system.

References

DS325, *OPB External Peripheral Controller Product Specification.*

SMSC LAN 91C111 32/16/8-Bit Three-In-One Fast Ethernet Controller Technical Reference Manual.

Data sheet of LAN91C111 10/100 Non-PCI Ethernet Single Chip MAC + PHY.

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
7/24/06	1.0	Initial Xilinx release.
9/25/06	1.1	Updated for EDK 8.2.01i.
6/5/07	1.2	Updated for EDK 9.1.01i.