Ethernet Controller

For Quartus II 13.0

1 Core Overview

The Ethernet Controller IP Core provides an interface between a Nios II processor and the Davicom DM9000a ethernet controller chip present on the DE2 and DE2-70 boards.

2 Functional Description

The Ethernet Controller IP Core provides access to the internal registers of the DM9000a chip. By reading and writing the registers, it is possible complete tasks such as initializing the chip, and sending and receiving packets. The core provides all required logic signals to and from the DM9000a chip, however users are required to supply a 25MHz clock to the DM9000a chip.

3 Instantiating the Core in Qsys

To instantiate the Ethernet Controller in Qsys, first locate the "Ethernet" core under University Program > Communications. Then add the core to your design and assign an address range to it. The core has no configurable parameters.

4 Software Programming Model

4.1 Register Map

The Ethernet Controller IP Core provides two memory mapped registers, as shown in Table 1. By using these two registers, the user can access all of the internal registers of the DM9000a.

Table 1. Ethernet Controller register map

Offset in bytes	Register name	Read/Write	Purpose
0	`INDEX`	W	The index (0x00 - 0xFF) of the internal register inside the DM9000a to access.
4	`DATA`	R/W	The data of the register at the specified index.

4.1.1 *INDEX* Register

By writing a value to this register, the user can specify which internal register of the DM9000a chip that they want to read or write. For example, the "RX Control Register" of the DM9000a is located at index 0x05. To write a value to this register, the user would write 0x05 to the *INDEX* register, then write the value to the *DATA* register. For the full list of the DM9000a internal registers and their indices, please refer to the DM9000a datasheet.

4.1.2 *DATA* Register

This register holds the value of the internal register specified by the *INDEX* register. Writing a value to this register will write the value to the internal register. Please refer to the DM9000a datasheet for a complete description of each internal register.

4.2 Programming with the Ethernet Controller

The Ethernet Controller core is packaged with C-language device drivers accessible through the hardware abstraction layer (HAL). The package offers a low level driver as well as a high level driver. The high level driver provides functions that initialize the chip, and send and receive packets. The low level driver provides functions that allow the user to read and write the internal registers of the DM9000a. To make effective use of the low level driver, the user should study the DM9000a documentation to learn the internal registers and their functions.

4.2.1 alt_up_ethernet_open_dev

Prototype:	`alt_up_ethernet_dev* alt_up_ethernet_open_dev( const char *name)`
Include:	`<altera_up_avalon_ethernet.h>`
Parameters:	`name` – the ethernet instance name. For example, if the instance name in Qsys is "ethernet_0", then *name* should be "/dev/ethernet_0"
Returns:	The corresponding device structure, or NULL if the device is not found.
Description:	Open the ethernet device specified by *name* .

4.2.2 alt_up_ethernet_init

Prototype:	`void alt_up_ethernet_init ( alt_up_ethernet_dev *ethernet)`
Include:	`<altera_up_avalon_ethernet.h>`
Parameters:	`ethernet` – struct for the ethernet device .
Description:	Initialize the DM9000a ethernet chip.

4.2.3 alt_up_ethernet_reg_read

Prototype:	`unsigned int alt_up_ethernet_reg_read( unsigned int base, unsigned int reg)`
Include:	`<altera_up_avalon_ethernet_low_level_driver.h>`
Parameters:	`base` – the base address of the ethernet device .
	`reg` – the index of the reg to read .
Returns:	The value of the register.
Description:	Read a control/status register of the DM9000a chip.

4.2.4 alt_up_ethernet_reg_write

Prototype:	`void alt_up_ethernet_reg_write( unsigned int base, unsigned int reg, unsigned int data)`
Include:	`<altera_up_avalon_ethernet_low_level_driver.h>`
Parameters:	`base` – the base address of the ethernet device .
	`reg` – the index of the reg to write to .
	`data` – the data to write.
Description:	Write to a control/status register of the DM9000a chip.

4.2.5 alt_up_ethernet_phy_read

Prototype:	`unsigned int alt_up_ethernet_phy_read( unsigned int base, unsigned int reg)`
Include:	`<altera_up_avalon_ethernet_low_level_driver.h>`
Parameters:	`base` – the base address of the ethernet device .
	`reg` – the index of the phy reg to read .
Returns:	The value of the phy register.
Description:	Read a phy register of the DM9000a chip.

4.2.6 alt_up_ethernet_phy_write

Prototype:	`void alt_up_ethernet_phy_write( unsigned int base, unsigned int reg, unsigned int data)`
Include:	`<altera_up_avalon_ethernet_low_level_driver.h>`
Parameters:	`base` – the base address of the ethernet device .
	`reg` – the index of the phy reg to write to .
	`data` – the data to write.
Description:	Write to a phy register of the DM9000a chip.

4.2.7 alt_up_ethernet_send_pkt

Prototype:	`unsigned int alt_up_ethernet_send_pkt( alt_up_ethernet_dev * ethernet_dev, unsigned char * data_ptr, unsigned int pkt_length)`
Include:	`<altera_up_avalon_ethernet_high_level_driver.h>`
Parameters:	`ethernet_dev` – the ethernet device . `data_ptr` – pointer to the array of bytes representing the bytes of the packet . `pkt_length` – the length, in bytes, of the packet.
Returns:	`0`: packet sent . `-1`: sending timed out .
Description:	Send a packet containing the bytes pointed to by data_ptr.

4.2.8 alt_up_ethernet_receive_pkt

Prototype:	`int alt_up_ethernet_receive_pkt( alt_up_ethernet_dev * ethernet_dev, unsigned char * data_ptr, unsigned int *pkt_length)`
Include:	`<altera_up_avalon_ethernet_high_level_driver.h>`
Parameters:	`ethernet_dev` – the ethernet device . `data_ptr` – pointer to the array of bytes representing the bytes of the packet . `pkt_length` – the length, in bytes, of the packet.
Returns:	`0`: packet received . `-1`: rx fifo empty . `-2`: the chip was reset to recover from an unstable state . `-3`: the packet was bad and flushed out without reading .
Description:	Receive a packet and store the data in the array pointed to by data_ptr. The length of the packet is written to *pkt_length.

4.2.9 alt_up_ethernet_set_interrupts

Prototype:	`unsigned int alt_up_ethernet_set_interrupts( alt_up_ethernet_dev * ethernet_dev, unsigned int interrupts)`
Include:	`<altera_up_avalon_ethernet_high_level_driver.h>`
Parameters:	`ethernet_dev` – the ethernet device . `interrupts` – the interrupts to enable. A value of 1 will enable packet received interrupts. For other interrupts, please refer to the DM9000a datasheet, specifically the section on the "Interrupt Mask Register".
Returns:	0 on success.
Description:	Enable or disable interrupts. By default, interrupts are disabled upon initializing the DM9000a with the `alt_up_ethernet_init` function.