Interfacing the Am29200 RISC Microcontroller to the MACE™ Am79C940 Ethernet Controller

AMD

Application Note

by Phil Simmons

INTRODUCTION

This report outlines the interface between a Am29200 RISC microcontroller and a MACE Am79C940 Ethernet controller to address embedded applications. This solution targets system cost and shows a simple glueless interface that can significantly reduce hardware design time. The 29K series of microprocessors and microcontrollers are completely software compatible and hence enables a designer to visualize a range of products without having to re-invest major programming resources for every level of processor performance.

AM29200 RISC MICROCONTROLLER

The Am29200 microprocessor is the first in a line of RISC microcontrollers based on the 29K architecture that targets systems that do not require the extremely high performance of other 29K microprocessors, but needs very low system cost made possible by the integration of processor, certain peripherals and support logic. The on-chip functions include: a ROM controller, a DRAM controller, a Peripheral Interface Adapter, a DMA controller, a Programmable I/O Port, a Parallel Port, a Serial Port, a Video Interface, and an Interrupt Controller.

The solution shown in this application note provides a minimal interface between the Am29200 RISC microcontroller and the Am79C940 MACE Network Interface Controller using the DMA facilities integrated into the Am29200.

MACE AM79C940 ETHERNET CONTROLLER

The Media Access Controller for Ethernet (MACE) is a LAN controller designed for a system with centralized or system specific DMA facilities. The controller has a high speed 16 bit slave register based interface where all transfers to or from the system are performed using simple read and write cycles.

The MACE provides an IEEE 802.3 interface that may be tailored to a specific application. It provides a complete Ethernet node solution with integrated 10BASE-T transceiver and supports up to 25 MHz system clocks. The Am79C940 embodies the Media Access Control (MAC) and Physical Layer Signaling (PLS) sub-layers of the IEEE 802.3 standard, and provides an IEEE defined Attachment Unit Interface (AUI) for coupling to an external Medium Attachment Unit (MAU).

Additional features also enhance over-all system design. The individual transmit and receive FIFOs optimize system overhead, providing substantial latency during packet transmission and reception, and minimizing intervention during normal network error recovery. The integrated Manchester encoder/decoder eliminates the need for an external Serial Interface Adapter (SIA) in the node system.

HARDWARE DESIGN

The objective of this design was to produce a processor-ethernet controller interface requiring minimal additional support logic, but also providing the performance expected from ethernet.

Memory Support

The memory support required to implement this ethernet engine is both ROM and RAM based. The ROM of choice is the Am29F010 Flash EPROM, although read-only PROMs are equally applicable if field code updates are not necessary. This 5 V only device provides 128 KBytes of memory used primarily for instructions and parameters. The organization of this device is 8 bits wide and if the designer arranges 4 devices together to produce a 32 bit wide image then 512 KBytes provides both sufficient space to run real-world applications as well as the 32 bit bus width that provides an efficient code interface for the processor. If 90 ns devices are used, then the processor will perform 2 cycle access to the ROM.

DRAM is required as the destination of the ethernet packets from the network. The Am29200 currently provides an interface that will burst 3 cycles for an initial access to a page, followed by 2 cycles for all subsequent accesses to that page. This uses readily available 100 ns fast page mode DRAMs and requires no support logic for control.

DMA Options

The Am29200 provides either hardware or software options for producing a DMA engine that will transfer an ethernet packet between the MACE and DRAM.

Software DMA

The 29K family of RISC microprocessors provides two special instructions, load multiple (LOADM) and store multiple (STOREM). These instructions provide the ability to transfer up to 192 data words between on-chip

Publication# **20315** Rev. **A** Amendment **/0**
Issue Date: **August 1995**

registers and memory structures with a single command. This can produce a highly efficient DMA engine under complete control of the software kernel, but can also be interrupted by a higher priority task.

Hardware DMA

The Am29200 provides two on-chip DMA channels. The designer can configure one channel to receive packets and the other to transmit packets (Note that a printer application does not necessarily require a DMA channel for transmit as almost all traffic is received and a transmit packet can be pre-loaded into the large MACE FIFO before being instructed to transmit).

In order to give minimum code development time while meeting the performance requirements of ethernet full-wire speed (see Performance), the design uses the hardware DMA channels of the Am29200. DMA channel 0 is set for the receive queue and DMA channel 1 for the transmit queue. When a frame is ready to be transmitted, the DMA channel is configured to transfer the complete frame less one 16-bit word. When the transfer has completed, the last word is written to the MACE FIFO by a simple STORE command with the EOF flag set to indicate end of frame. The receive transfer count is set to maximum and runs freely until the frame is completely received into DRAM at which point the MACE will indicate that the last byte of the frame has been transferred with the EOF signal which terminates the DMA by connecting to the TDMA input to the Am29200 DMA controller.

MACE Interface

The MACE is set to run at 16 MHz even though the device can operate at 25 MHz. This is so that the interface is synchronous and so that no external logic is required. The SCLK for the MACE is therefore driven from the Am29200 MEMCLK output. The MACE can be programmed to operate from either the rising or falling edges. The Am29200 requires valid data with respect to rising clock edges and therefore EDSEL is tied to ground on the MACE.

The MACE is connected to the peripheral port of the Am29200. As DMA channel 0 and 1 are used and are both connected to the FIFO chip select for the MACE (FDS), the PIACS (0,1) signals must be ANDed to provide the FIFO select, and the register select for the MACE (CS) can be directly connected to PIACS2.

Even though there is an output enable from the Am29200 specifically for peripheral interfaces, the normal read/write signal (R/W) is used to meet timing requirements.

The RDTREQ and TDTREQ DMA requests issued by the MACE are connected to the DREQ (1-0) signals of the Am29200 (Channel 0 for receive and channel 1 for transmit queues). End of Frame (EOF) is connected to TDMA though an inverting gate for receive cycles, and from a PIO output through an open collector gate for transmit cycles.

Ethernet Interface

The MACE incorporates both AUI and 10BASE-T standard interfaces. The AUI interface can be used to provide 10BASE-2 connectivity by using an external transceiver. The 10BASE-T interface simply requires the external filter/transformer and 6 resistors for protection of the MACE from the twisted-pair connection to the outside world.

System Power Requirements

The system power requirements are especially low for this design as the Am29200 is highly integrated and the addition of support logic is minimal. The MACE incorporates three modes of sleep, and can be remotely or automatically awoken. When the MACE is completely asleep the device typically requires 10 µA, when dozing requires typically 3 mA, and when fully operational requires typically 40 mA. The Am29200 can initiate power savings for the MACE by register updates and hardware control of the SLEEP pin which is connected to a PIO output signal.

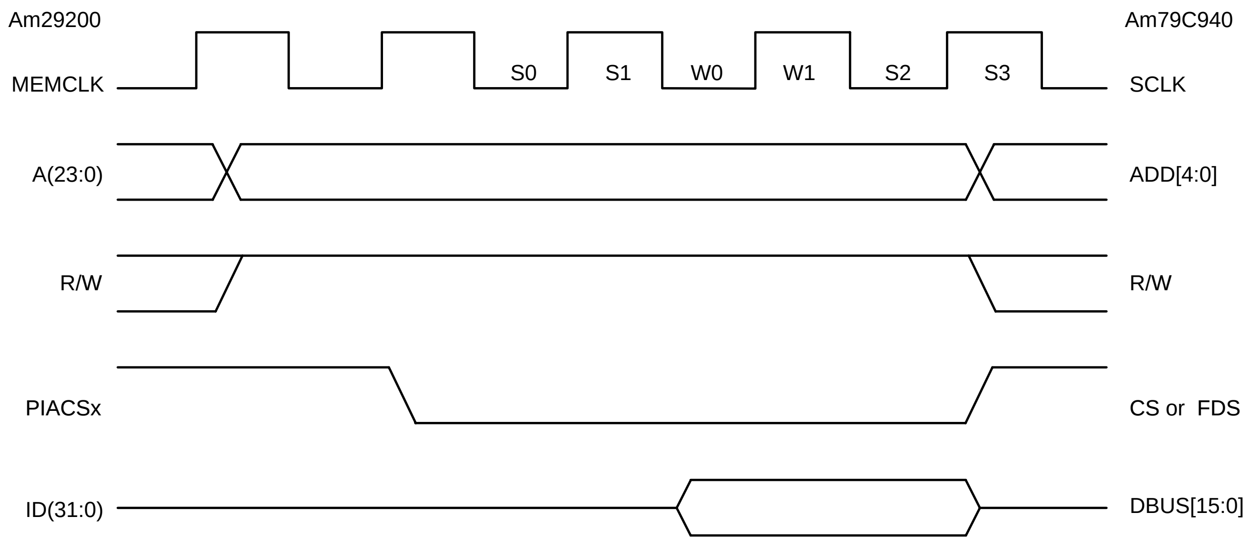
Timing Diagrams

The synchronous cycles required for transfers is programmable for both the Am29200 and the MACE devices.

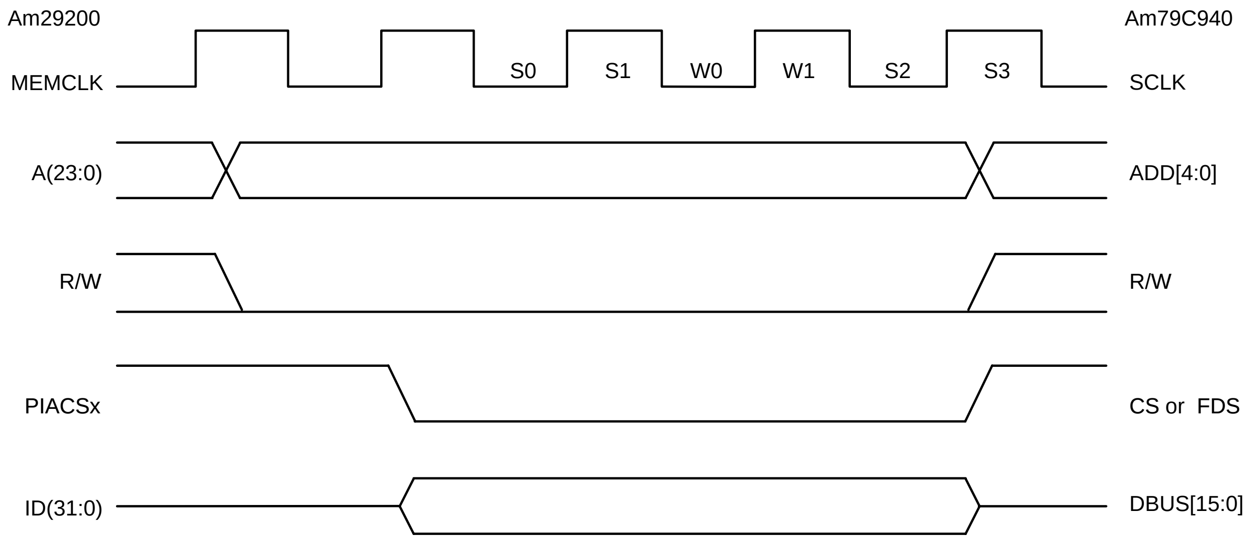
The peripheral interface on the Am29200 that the MACE connects to is programmed using software. An Am29200 peripheral read cycle can be achieved in 3 clocks and a write cycle in 4 clocks. This design therefore sets the transfer rate as 4 clocks for the Am29200 so that read and write transfers are the same.

The transfer timing for the MACE is determined by the status of an external pin (TC). This is tied to ground so that the MACE is configured to transfer in 3 clock cycles. The number for the Am29200 and the MACE differ because the initial Am29200 clock cycle is used to establish addresses before asserting the appropriate peripheral chip select.

Read Cycle



Write Cycle



SOFTWARE

This article describes some details for configuring a MACE and the basic philosophy for transmission and reception of packets.

Initialization

The MACE will power-on with its registers set to the default values. The initialization sequence will configure the bus interface, set the FIFO and interrupt configuration, set the ethernet interface configuration, run diagnostics, and then configure for online operation.

Bus Interface (BIUCC)

The Byte Swap is set to big endian and the Transmit Start Point is set to 112 bytes so that the transmit FIFO is as full as possible before transmission commences.

FIFO Configuration (FIFOCC)

The Transmit and Receive FIFO Watermarks are set to the minimum values (XMTFW = 8 cycles, RCVFW = 16 bytes) so that the processor can commence DMA as soon as possible. The update bits are set, in order that the watermarks can be modified. The Transmit and Receive Burst flags are set to enabled these functions.

Interrupt Mask (IMR)

All interrupts are masked until initialization is complete.

PLS Configuration (PLSCC)

The 10BASE-T port is selected, but it can be overridden as the PHYCC register will set automatic port selection.

PHY Configuration

The Link Test, Auto Polarity Correction, and Auto Port Selection are all enabled.

Internal Address (IAC)

Set the Address Change. If multicast addressing is permitted, then the LOGADDR bit is set and the access is followed by 8 write cycles to the Logical Address register (LADRF), else the PHYADDR bit is set and the access is followed by 6 writes to update the Physical Address register (PADR).

Logical or Physical Addressing

The Logical Address Register requires a 64 bit mask, and is updated by performing 8 bytes writes, least significant to most significant. The Physical Address Register is accessed in the same manner except only 48 bits are required through 6 byte write cycles.

MAC Configuration (MACCC)

The transmit and receive functions are enabled by accessing this register.

PERFORMANCE

Ethernet performance is normally defined by the ability to transfer 64 byte ethernet packets at full wire speed as this requires highest system performance.

Ethernet will transfer packets at 10 Mb/s. For this example the minimum packet size is defined as 72 bytes (64 bytes of information and 8 bytes of synchronization) and the interpacket gap will be 9.6 µs. Using these parameters the CPU must process 14880 packets every second.

Minimum Size Packet CPU

Ethernet Packet			IEEE 802.3 Packet		
	Bits	**Bytes**		**Bits**	**Bytes**
Preamble	62	7.75	Preamble	56	7
Synchronization	2	.25	Synchronization	8	1
Destination Address	48	6	Destination Address	48	6
Source Address	48	6	Source Address	48	6
Type	16	2	Length	16	2
Data	368	46	Data	368	46
Checksum	32	4	Checksum	32	4
Total	576	72	Total	576	72
Time (µs) @ 10Mb/s	57.6				
Inter-Packet-Gap (µs)	9.6				
Total Packet Time (µs)	67.2				
Packets per second	14880				

Bus Utilization

Bytes to be transferred to/from Host (includes header and frame status)	64
Transfers for 16 bit data bus (4 bytes frame status requires byte access)	34
Clocks per DMA transfer MACE read/write	4
Clocks per DMA transfer DRAM read/write	3
Total Clocks per DMA transfer (includes 1 clock DMA turn around)	8
Total Clocks to DMA minimum size packet	272
Total Time to DMA minimum size packet @ 16 MHz (60 ns)	16.32 µs
Percentage Bus Bandwidth required for packet DMA	**24.29%**
Time remaining following minimum size packet transfer	50.88 µs
Number of Clocks per instruction (execution from FLASH memory)	2
Time per 29200 instruction	120 ns
Number of instructions that can be executed in 50.88 µs	424

As all Am29200 instructions execute in a single clock cycle (except LOADM, STOREM, and IRET) we can estimate the number of instructions available to perform a real world application while still operating at full ethernet wire speed.

The 424 instructions figure is a quick estimate that can be expanded to include such things a interrupt latency, or refresh overhead, and gain an accurate number of instructions available to the programmer while writing driver code.

CONCLUSIONS

The design shown here displays the ease of developing real world solutions with the Am29200 microcontroller. Performance can be quickly estimated due to the single clock per instruction execution unit for all members of the 29K family.

```
;PALASM Design Description

;-----------------Declaration Segment------
TITLE     Am29200–MACE control
PATTERN   A
REVISION  1.0
AUTHOR    Phil Simmons
COMPANY   AMD
DATE      02/21/93

CHIP _200_mace PALCE16V8

;------------------PIN Declarations--------
PIN  2    /PIACS0                      ; INPUT
PIN  3    /PIACS1                      ; INPUT
PIN  4    /TDTREQ                      ; INPUT
PIN  5    /RDTREQ                      ; INPUT
PIN  6    PIO1                         ; INPUT
PIN  7    RW                           ; INPUT
PIN  8    /RESET                       ; INPUT
PIN  19   /FCS                         ; OUTPUT
PIN  18   DREQ0                        ; OUTPUT
PIN  17   DREQ1                        ; OUTPUT
PIN  16   /EOF                         ; IO
PIN  15   TDMA                         ; OUTPUT

;----------Boolean Equation Segment--------
EQUATIONS

FCS   = PIACS0 * /RESET
      + PIACS1 * /RESET

DREQ0 = RDTREQ * /RESET

DREQ1 = TDTREQ * /RESET

EOF      = /PIO1  * /RESET
EOF.TRST = PIACS1 * /RESET

TDMA = PIACS0 * RW  * EOF * /RESET

;-------------Simulation Segment-----------
SIMULATION

TRACE_ON EOF TDMA DREQ0 DREQ1 FCS
;initialisation
SETF  /PIACS0 /PIACS1 /TDTREQ  /RDTREQ  PIO1  RW   RESET
SETF  /PIACS0 /PIACS1 /TDTREQ  /RDTREQ  PIO1  RW   RESET
SETF  /PIACS0 /PIACS1 /TDTREQ  /RDTREQ  PIO1  RW   /RESET
SETF  /PIACS0 /PIACS1 /TDTREQ  /RDTREQ  PIO1  RW   /RESET
SETF  /PIACS0 /PIACS1 /TDTREQ  /RDTREQ  PIO1  RW   /RESET
SETF  /PIACS0 /PIACS1 /TDTREQ  /RDTREQ  PIO1  RW   /RESET
SETF  /PIACS0 /PIACS1 /TDTREQ  /RDTREQ  PIO1  RW   /RESET

;receive data
SETF  /PIACS0 /PIACS1 /TDTREQ  RDTREQ   PIO1       RW  /RESET
SETF  /PIACS0 /PIACS1 /TDTREQ  RDTREQ   PIO1       RW  /RESET
SETF  PIACS0  /PIACS1 /TDTREQ  RDTREQ   PIO1  /EOF RW  /RESET
SETF  PIACS0  /PIACS1 /TDTREQ  RDTREQ   PIO1  /EOF RW  /RESET
SETF  PIACS0  /PIACS1 /TDTREQ  RDTREQ   PIO1  /EOF RW  /RESET
SETF  PIACS0  /PIACS1 /TDTREQ  RDTREQ   PIO1  /EOF RW  /RESET
SETF  /PIACS0 /PIACS1 /TDTREQ  RDTREQ   PIO1       RW  /RESET

;receive last byte of frame
SETF  PIACS0  /PIACS1 /TDTREQ  RDTREQ   PIO1  EOF  RW  /RESET
SETF  PIACS0  /PIACS1 /TDTREQ  RDTREQ   PIO1  EOF  RW  /RESET
SETF  PIACS0  /PIACS1 /TDTREQ  RDTREQ   PIO1  EOF  RW  /RESET
SETF  PIACS0  /PIACS1 /TDTREQ  RDTREQ   PIO1  EOF  RW  /RESET
SETF  /PIACS0 /PIACS1 /TDTREQ  /RDTREQ  PIO1       RW  /RESET
SETF  /PIACS0 /PIACS1 /TDTREQ  /RDTREQ  PIO1  RW    /RESET
SETF  /PIACS0 /PIACS1 /TDTREQ  /RDTREQ  PIO1  RW    /RESET

;transmit data
SETF  /PIACS0 /PIACS1 TDTREQ   /RDTREQ  PIO1  RW    /RESET
SETF  /PIACS0 /PIACS1 TDTREQ   /RDTREQ  PIO1  RW    /RESET
SETF  /PIACS0 PIACS1  TDTREQ   /RDTREQ  PIO1  /RW   /RESET
SETF  /PIACS0 PIACS1  TDTREQ   /RDTREQ  PIO1  /RW   /RESET
SETF  /PIACS0 PIACS1  TDTREQ   /RDTREQ  PIO1  /RW   /RESET
SETF  /PIACS0 PIACS1  TDTREQ   /RDTREQ  PIO1  /RW   /RESET
SETF  /PIACS0 /PIACS1 TDTREQ   /RDTREQ  PIO1  RW    /RESET
SETF  /PIACS0 /PIACS1 TDTREQ   /RDTREQ  PIO1  RW    /RESET

;transmit last byte of frame
SETF  /PIACS0 PIACS1  TDTREQ   /RDTREQ  /PIO1 /RW   /RESET
SETF  /PIACS0 PIACS1  TDTREQ   /RDTREQ  /PIO1 /RW   /RESET
SETF  /PIACS0 PIACS1  TDTREQ   /RDTREQ  /PIO1 /RW   /RESET
SETF  /PIACS0 PIACS1  TDTREQ   /RDTREQ  /PIO1 /RW   /RESET
SETF  /PIACS0 /PIACS1 /TDTREQ  /RDTREQ  PIO1  RW    /RESET
SETF  /PIACS0 /PIACS1 /TDTREQ  /RDTREQ  PIO1  RW    /RESET
SETF  /PIACS0 /PIACS1 /TDTREQ  /RDTREQ  PIO1  RW    /RESET
SETF  /PIACS0 /PIACS1 /TDTREQ  /RDTREQ  PIO1  RW    /RESET
SETF  /PIACS0 /PIACS1 /TDTREQ  /RDTREQ  PIO1  RW    /RESET
TRACE_OFF
;------------------------------------------
```

This page contains schematics which can be found in the NPD data book PID#14287D

This page contains schematics which can be found in the NPD data book PID#14287D

This page intentionally left blank.

This page contains schematics which can be found in the NPD data book PID#14287D

This page contains schematics which can be found in the NPD data book PID#14287D

This page contains schematics which can be found in the
NPD data book PID#14287D

This page contains schematics which can be found in the NPD data book PID#14287D

This page contains schematics which can be found in the NPD data book PID#14287D

This page contains schematics which can be found in the NPD data book PID#14287D