Interfacing Fast Ethernet to Processors

By Mike Jones
Senior FAE, Micrel Inc.

The roll out of low cost broadband services has led to an 'IP' network revolution across factory, office and home. Ethernet has almost exclusively become the physical layer for all networking applications, whether they be industrial control, office routers or consumer equipment such as Home Gateways, VoIP phones or Set-Top-Boxes.

Ethernet has dominated due to the ease-of-use and open standard approach providing true interoperability between equipment in the field. By itself, Ethernet is not strictly a protocol yet it provides the lower 'hardware' layers for the higher layer software stacks such as TCP/IP or UDP. So, it would seem the task to network a device should be as simple as 'bolting on' an Ethernet transceiver and modifying the software. Indeed, this is basically true, however, the many different interfaces offered by today's vast number of Processors and FPGAs often adds uncertainty.

So how do I interface my Ethernet PHY (transceiver), multi-port switch or controller to my chosen processor? This depends firstly, if the processor provides an integrated MAC (Media Access Control).

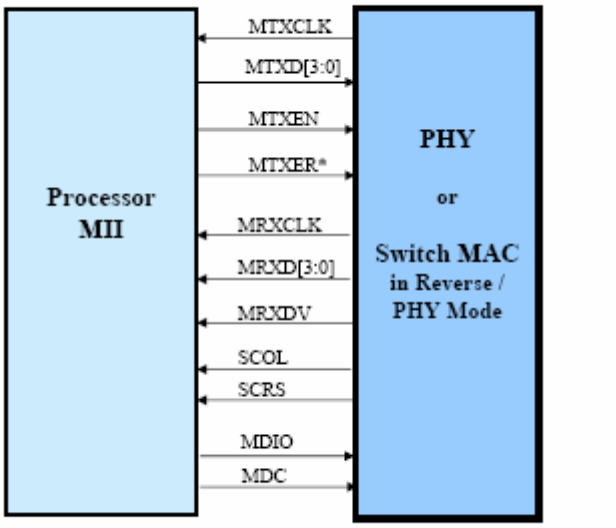
Interfacing to Processors with Integrated MAC

The two basic building blocks in Ethernet are the MAC (controller) and the PHY (transceiver). The MAC-PHY interface comprises of two signal groups; a data bus and a management bus, and is known as the MII (Media Independent Interface). The MII is specified in clause 22 of the IEEE 802.3 standard. This interface is independent of processor and physical media (copper or fibre), providing both simplicity and interoperability.

MII Interface

MII comprises of 4-bit transmit, TXD[3:0], and receive data, RXD[3:0], buses operating from independent 25MHz clocks TXCLK (local clock) and RXCLK (recovered line clock), respectively. Additionally, on the transmit side, TXEN indicates when data is sent by the MAC and TXER indicates if errors have occurred during transmission. Similarly, the receive side uses RXDV to indicate valid data and RXER to signal if physical layer errors are detected. For half duplex operation signal COL indicates a collision during transmission, and CRS signals the presence of data transmission and/or data reception.

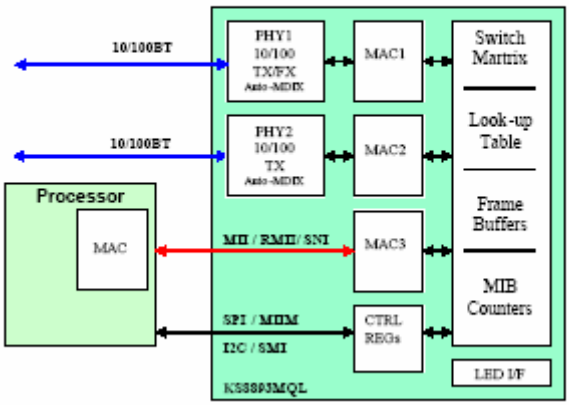
Figure 1. MII Interface between Processor and Ethernet PHY or Switch



* Tie TXER input pin to ground if MAC does not support transmit error

When interfacing a multi-port Ethernet switch to a processor using MII then the connectivity is not as obvious due to the MAC-MAC configuration, as shown in the example in Figure 2. The solution is for the Ethernet switch to act like a 'PHY', by sourcing both TXCLK and RXCLK. Here the switch is configured to what is known as Reverse or PHY mode.

Figure 2. Interfacing a Multi-Port Switch to Processor using MAC-MAC Configuration



The MII interface is such that often configuration can be provided by strap-in pins without the need for a device driver. The switch or PHY can operate in a standalone mode independent from the processor. If management is required then the MIIM (Media Independent Interface Management) bus is defined also in IEEE 802.3 clause 22 to access a set of standard MIIM PHY registers and customized registers. Hence, the basic PHY operation provided by an Ethernet driver is vendor independent. Only access to the customized vendor specific registers will differentiate the code. Switch configuration registers are almost always vendor specific and may be provided either via the MIIM or other management interfaces, such as I2C or SPI.

RMII and SMII Interfaces

To reduce the pin count for multi-port FPGA and ASIC designs a vendor led consortium introduced the RMII™ Specification (Reduced MII). RMII provides independent 2-bit wide transmit and receive paths synchronised to a common 50MHz reference clock. Furthermore, carrier sense and receive data valid are now combined as one signal CRS_DV. Collision detection is regenerated on the MAC side by ANDing signals TX_EN and CRS_DV. This reduces the total pins per port to 8 from 16 (MII) as shown in Figure 3 below. Figure 3 also illustrates that architecturally, RMII is not a replacement for the MII, but an additional reconciliation layer on either side of the MII.

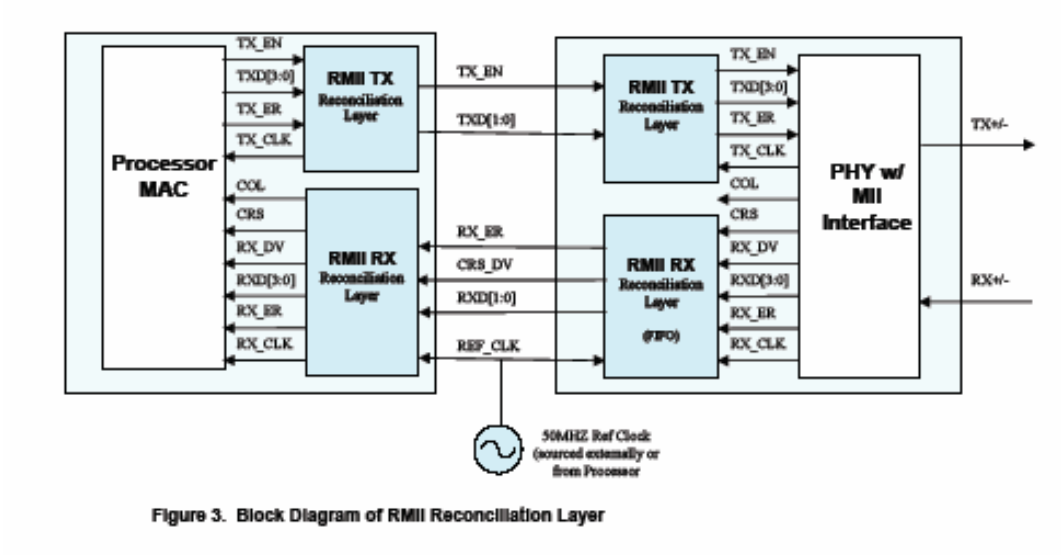


Figure 3. Block Diagram of RMII Reconciliation Layer

The Serial Media Independent Interface (SMII) provides the lowest pin count Media Independent Interface with a total of only 4 pins; a 125MHz reference clock, 12.5MHz SYNC signal generated by the MAC, and single-bit receive and transmit data. SMII is unlike MII and RMII in its approach as the control signals are embedded into the receive and transmit data streams. The SYNC signal is used to frame ten bit segments where the first two bits are carrier sense (CRS) and receive data valid (RX_DV) or transmit error (TX_ER) and transmit enable (TX_EN) in the receive and transmit directions respectively.

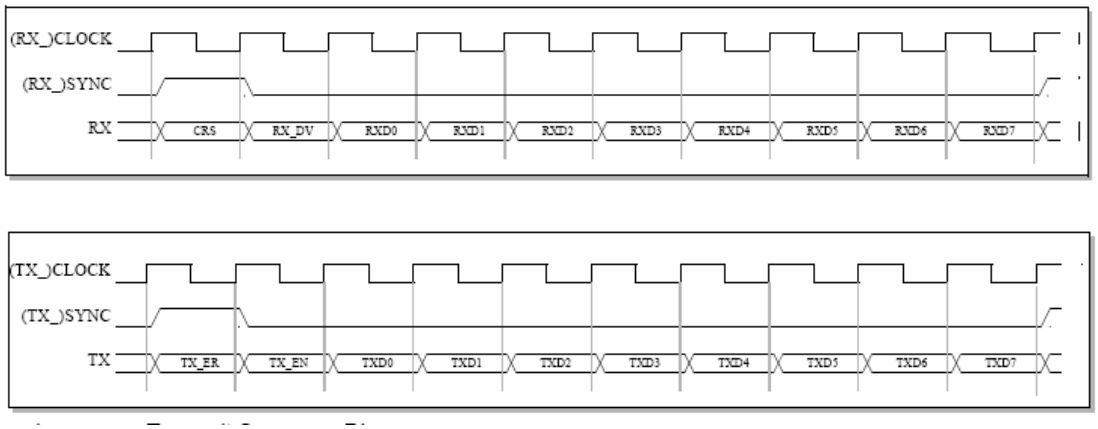


Figure 4. SMII Receive and Transmit Data Sequence Diagrams

Which Media Independent Interface Should One Choose?

Of course much depends upon which of the MII interfaces the chosen processor supports. Although not as common as MII the RMII and SMII would seem the most obvious preferred interfaces. However, both RMII and SMII do have two distinct

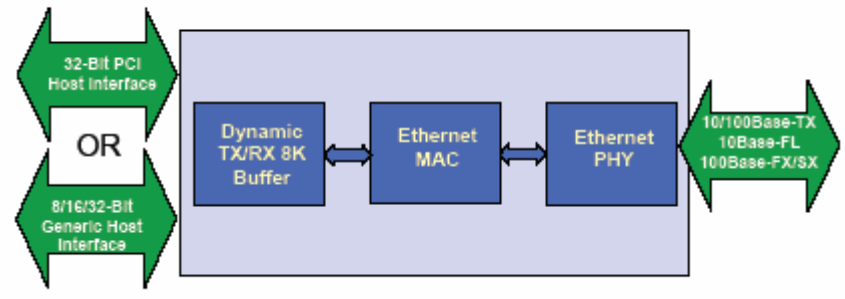
disadvantages. Firstly, by operating at faster data rates (50Mbps and 125Mbps) then there is likely to be a bigger impact on EMC emissions, compared to the slower 25Mbps MII. The MII interface should also be used in real-time applications. RMII and SMII provide both receive and transmit data paths synchronous to the external reference clock, unlike MII. Transferring the recovered incoming line clock to the reference clock requires a FIFO to tolerate differences in frequency. Hence, traffic latency is increased and will vary with reference and line clock frequency drift. Variable delays are far more critical than fixed delays in a real-time network, as they are unknown and cannot be compensated for. The Ethernet specification calls for a reference clock frequency tolerance of maximum 100ppm (0.01%). However, it is not uncommon to encounter frequency errors up to 0.1%, in today's network. Therefore, the required FIFO size needs to be a minimum 27 bits, introducing latency jitter of 27 x 10nS = 270nS for SMII and RMII interfaces over MII.

FIFO Size (bits) = 2 x (Max Frame Size) x (Network Error + Local Error)
= 2 x (1518 x 8) x (0.1% + 0.01%)
= 26.7 bits.

Interfacing to Processors without integrated MAC

Ethernet applications are not limited to just networking processors with integrated MAC. Indeed many of today's embedded applications, for example Video and VoIP phones and Set Top Boxes do not use such kind of processors. To interface Ethernet to a processor without integrated MAC, requires an Ethernet Controller supporting PCI or Generic Host bus, as demonstrated in the simplified KSZ8841 block diagram of Figure 5.

Figure 5. Simplified KSZ8841 Ethernet Single-Port Controller Block Diagram



Non PCI Ethernet Controllers offer great flexibility allowing glueless interfacing to virtually all 8, 16 or 32-bit processors. Figure 6 shows an example of interfacing Micrel's KSZ8841-16MQL Single-Port Ethernet Controller to the Renesas M16C 16-bit processor, using asynchronous (ISA-like) transfer mode. This is perhaps the most common transfer mode implementation. However, asynchronous EISA-like or VLBus-like and synchronous mode is available depending on the needs of the processor.

PCI controllers will usually provide higher throughput performance over the non-PCI controllers. True wire-speed 100Mbps may be possible, although much will depend on the available CPU processing power, rather than the controller hardware. The KSZ884x family enhances performance by offering a true dynamic buffering scheme for packet data transfer between controller and processor. This scheme optimises the buffer resource by dynamically allocating the minimum required memory for the packet size.

Unlike the PHY or switch with MII, an Ethernet Controller will need software intervention to operate in the form of a driver.

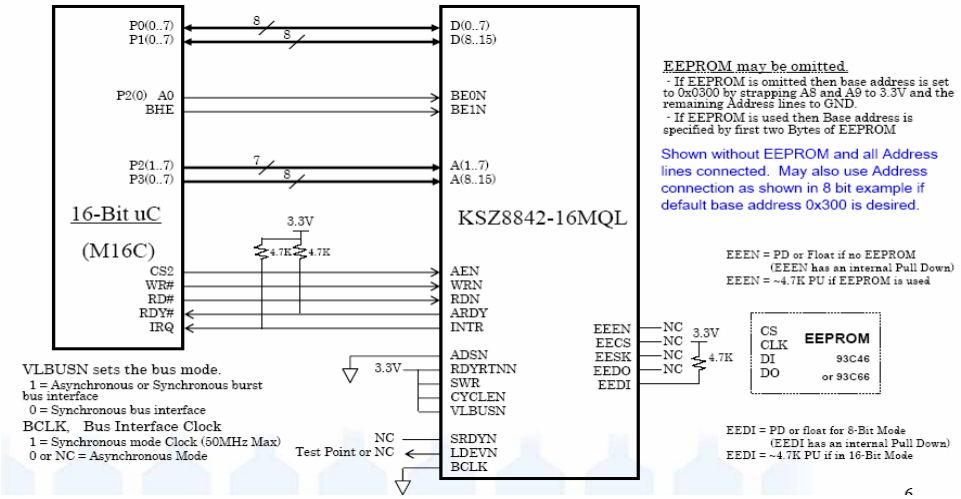


Figure 6. KSZ8842-16MQL 2-Port Ethernet Controller Interfacing to a 16-Bit Processor

Dual Port Ethernet applications can be realised with the unique KSZ8842 family of PCI and non-PCI controllers offering a single-chip solution which is pin compatible with their Single-Port counterpart. A cost effective, common PCB design can now be designed, as demonstrated in Figure 7.

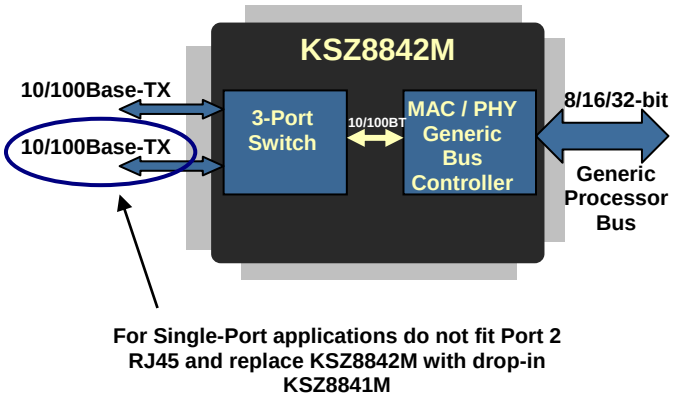
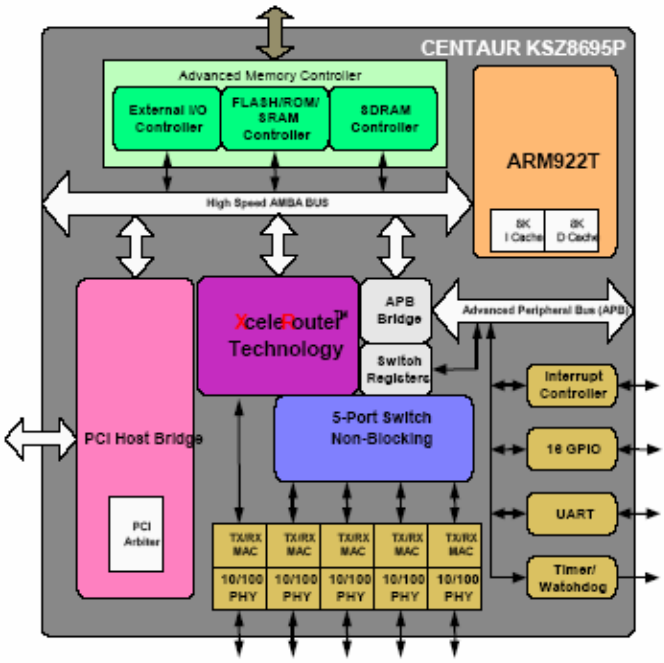


Figure 7. KSZ8842M Single-Chip Solution with Pin Compatible KSZ8841M.

System-On-Chip

An alternative approach to interfacing Fast Ethernet to a processor is to integrate both blocks into a single System-on-Chip (SoC). An example of such a device is Micrel's KSZ8695P shown in Figure 8 below. The ARM9 processor and 5-Port Ethernet switch are integrated into a single chip along with other common interfaces like PCI, UART, GPIO and memory controller.

Figure 8. KSZ8695P System-On-Chip Block Diagram



Obvious advantages of reduced real estate, power consumption and cost can be gained using such an approach.

Summary

The choice of processor in any application is critical for performance and functionality, as well as development costs. Most design teams will have invested considerable time and money into their expertise and tools for a specific processor family. It would be unrealistic to alter the choice of processor just to fulfil the demand for networking. It is clear the need for Ethernet to fit into the processor design, rather than the opposite. Micrel provides a 'One Stop Shop' for network enabling any design, whether it requires a PHY, switch, controller or even SoC solution. For further details on Micrel Ethernet Devices go to: http://www.micrel.com/products.do

###