Benefits from embedded protocol processing in an FPGA

by Sid Mohanty, Lattice Semiconductor

The marriage between FPGA and ASIC technology can yield significant benefits providing an embedded function that utilizes the flexibility of the FPGA.

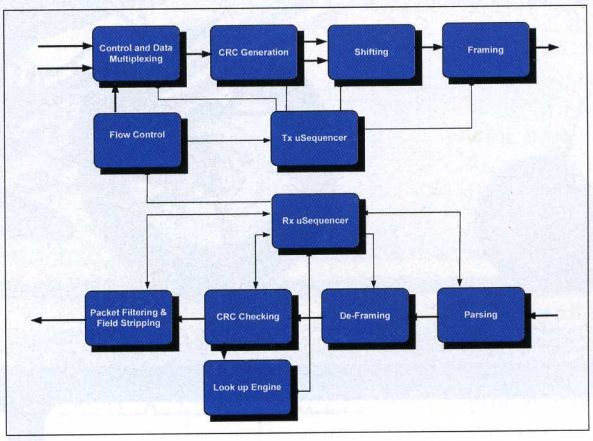

Figure 2. flexiMAC architecture

The move to volume manufacturing in 90 nm technologies has greatly increased the performance and density of FPGAs. At the same time, the soaring mask costs have made ASIC solutions uneconomic in 90 nm for anything but very high volume markets. Although this has narrowed the gap in performance, there is still a significant penalty in silicon area and power dissipation when selecting an FPGA solution. To address this problem, FPGA vendors are incorporating increasing amounts of ASIC technology with their programmable logic. Embedded hard cores such as multipliers, processors, transceivers and memories account for about 50% of many high specification FPGAs. However, the decision as to what to embed as a hard core is not always a simple one. For applications not using the ASIC function, this is effectively wasted area and adds to the cost of the FPGA.

The challenges posed by the various communications protocols is one area where the need can be identified for an ASIC type of hard core that can handle several differing communications standards. A hard core, co-existing with FPGA technology on the same chip, can offer design engineers an excellent marriage between the two technologies. Lattice Semiconductor has embedded an ASIC hard core termed "flexiMAC" into the LatticeSCM 90 nm family of FPGAs. This family of devices offers all the advantages of highly dense ASIC technology (i.e. low cost, low power and high performance), while retaining the flexibility to address different applications.

In essence, the flexiMAC is a simple packet processor core tailored to address standards that exploit the LatticeSC's high speed serial I/O. The standards are supported through a microcode programming architecture. Initially, Lattice has developed microcode for 1G Ethernet MAC, 10G Ethernet MAC, and PCI Express data link layer and physical layer functions for x1, x2 and x4 lane channels. The heart of the flexiMAC is a microcoded processor controlling a datapath customized to packet processing functions such as framing, CRC generation and checking, packet parsing, filtering and flow control.

These functions complement the physical layer functions for serial protocols implemented in the LatticeSC's embedded SERDES and physical coding sublayer (PCS) blocks. The flexiMAC is implemented in high density ASIC logic of ~50K gates so it occupies only a very small die area and saves up to 10x the silicon area over an FPGA LUT-based implementation of the same functions. The area saving is greatest for 10 Gbit/s data rates in which data-path functions such as CRC checking need a 64-bit datapath width in order to run on the FPGA fabric, making it very resource-intensive. For example, in 10G Ethernet mode there is a resource saving of approximately 6K LUTs. Figure 2 shows how the flexiMAC can be used to reduce system cost. In this case a 4-port 10 Gbit Ethernet switch is ported from an 80K LUT device with the 10G Ethernet MACs in soft logic, to a 40K device with Ethernet MACs implemented in two flexiMACs.

The following sections describe the architecture of the flexiMAC microsequencer. Figure 1 shows the architecture of the flexiMAC packet processor. The flexiMAC processes both the transmit and receive path of a given end-node. Any interaction between the transmit and receive path is controlled in the flow control block that is responsible for data-link layer flow control behaviour such as sending acknowledgements or pausing the transmission of packets. Both the transmit and receive sides are controlled by a microcoded microsequencer that is described in more detail later.

The following describes the transmit and receive paths of the flexiMAC. On the transmit side, packets are either transported from a data stream, a flow control block, or can be constructed from programmable registers within the "control and data multiplexing block". For example, this block can construct Ethernet PAUSE frames or define PCI Express initialization packets. The source of the transmitted packets is controlled by the microsequencer.

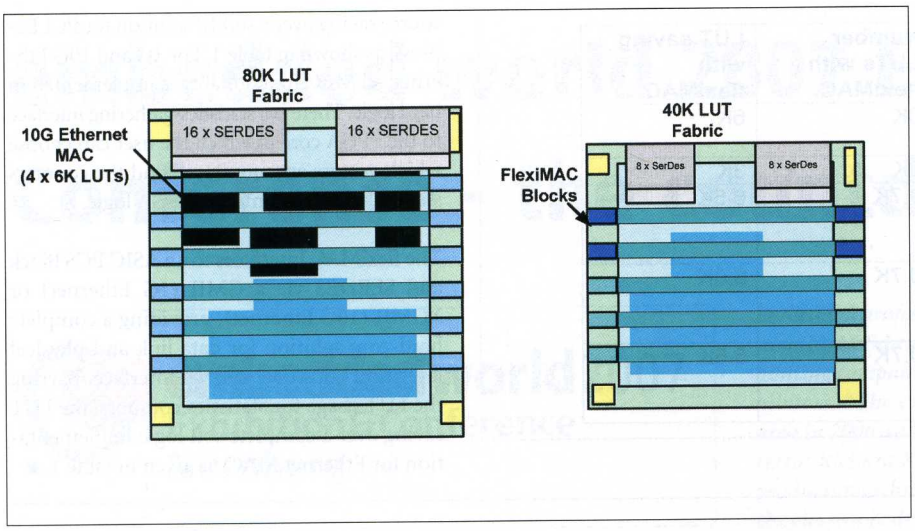

Figure 2. 10G Ethernet cost reduction via flexiMAC

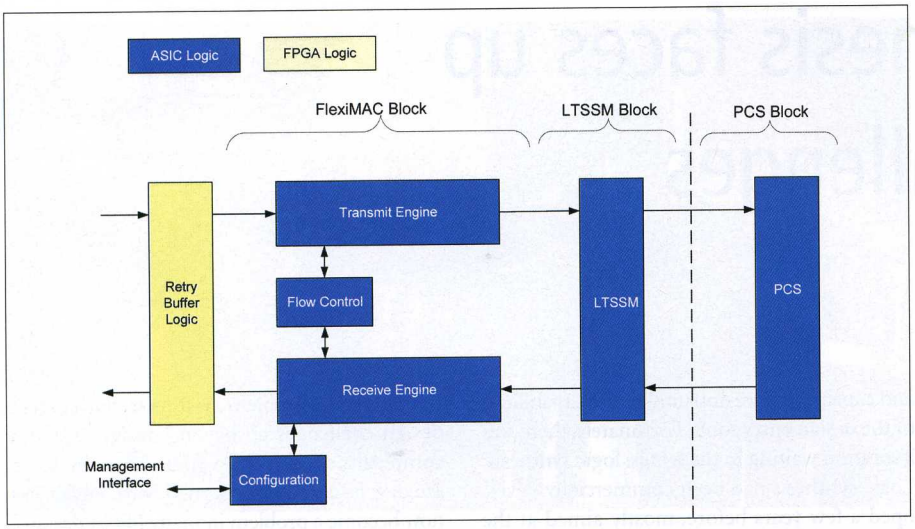

Figure 3. PCI Express data link layer and PCS partitioning

Packets from the control and data multiplexing block have the appropriate CRC calculated and appended onto the end of the packet. The CRC polynomial is selected by the microsequencer, depending on the type of packet being transmitted. The "Shifter" block shifts the data and CRC appropriately so that the packet can be framed and CRC appended in the correct position in the 32-bit data path.

For example, Ethernet packets have variable lengths, so the last byte position must be detected and the 32-bit CRC must be shifted to append correctly to the packet. Finally, the "framing" block appends or prefixes the appropriate bytes to the packet for framing and inserts "idles" or other control characters where appropriate. All this is controlled by the microsequencer and different frame characters can be programmed for alternate standards supported. Different standards are supported by programming registers in the transmit datapath for control packet construction and framing and by creating the appropriate microcode in the microsequencer. A microassembler has been developed to generate microcode images. The receive path has a "parser" block that searches each received word for special programmable control characters and reports back to the microcontroller. The microcontroller can select which characters are to be detected on a byte- and bit-level and also has the ability to look for boolean combinations of matches using a programmable mask. This is used to detect the beginning and end of packets and also special control words such as flow control messages or preambles. Together with the microsequencer, this is used for protocol checking and detection of flow control packets. Errors are flagged if protocol violations are detected and flow control actions are initiated for different protocols.

The "de-framer" block removes any framing bytes and re-aligns the data to the word boundary for CRC checking. The CRC checker checks the integrity of the data and CRC and tells the microsequencer if there is an error. The microsequencer selects which CRC polynomial to check. In parallel to the CRC checking is a programmable look-up engine that decides if the packet is to be discarded by accessing programmable registers for both unicast and multicast packets. A hashing function for multicast Ethernet packets can be derived from the CRC checker over the multicast destination address.

Finally, the "packet filtering and field stripping" engine is used to remove filtered packets and control packets if directed by the microsequencer. It is also used to remove indicated fields such as padding bytes from short Ethernet packets. All this is controlled by the microsequencer. Different standards are supported by programmable registers in the receive datapath for control packet parsing and lookup and by creating the appropriate microcode. The combination of programmable microcode and registers in the data path implements a highly flexible packet processor capable of data link layer processing for data streams of speeds up to 10 Gbit/s. The high degree of programmability allows changes to be made as standards evolve or problems are found during hardware interoperability testing. It also allows OEMs to customize standards for proprietary protocols across internal interfaces. For example, customers may use special preamble characters in Ethernet for configuration of backplanes. Data link layer functions such as CRC checking and generation, framing and packet parsing are highly programmable and microcode could be written to support other packet protocols in the future. In PCI Express mode the flexiMAC provides partial data link layer and physical layer functions.

The flexiMAC, together with PCS, SERDES and soft logic, provides a complete data link layer and physical layer solution for x1, x2 and x4 lane PCI Express nodes. The flexiMAC hard logic provides the following PCI Express functions in ASIC logic: 1) framing of TLP and DLLP packets; 2) CRC checking and generation for TLP and DLLP packets; 3) flow control initialization packet for VC-0 by the generation and checking of FC-Init DLLP packets; 4) generation and detection of Ack and Nack DLLP flow-control packets and 5) insertion of SKIP characters. The LTSSM block is a separate block and implements the LTSSM state machines for a x4, x2 or x1 lane PCI Express link. The PCS provides the following PCI Express Physical Layer functions: 1) 8b/10b encoding and decoding; 2) scrambling; 3) clock compensation logic; 4) comma detection and 5) lane alignment.

Invoking a PCI Express node in the development software – ispLEVER, will instantiate both the hard and soft IP logic, to give a complete data link layer and physical layer endpoint solution for PCI Express. This gives a considerable re-

IP	Number LUTs Soft IP	Number LUTs with flexiMAC	LUT saving with flexiMAC
10G Ethernet MAC	6K	0K	6K
1G MAC	3K	0K	3K
x4 PCI-Express PHY & DLL	7.2K	0.7K	6.5K
x2 PCI-Express PHY & DLL	6.7K	0.7K	6.0K
X1 PCI-Express PHY & DLL	6.2K	0.7K	5.5K

Table 1. Resource usage saving using the flexiMAC

source saving over a soft IP solution for PCI Express, as shown in table 1. For 1G and 10G Ethernet, all MAC functionality is implemented in hard logic. There is a statistics-gathering interface to the FPGA core in which the user can choose which vectors are monitored and the counters should be implemented in FPGA logic.

The flexiMAC interfaces to an ASIC PCS block and SERDES via a GMII (1G Ethernet) or XGMII (10G Ethernet), providing a complete hard-core solution for data link and physical layer functions over a serial interface. Barring the LUT usage for statistics counters, the LUT saving over a complete soft logic implementation for Ethernet MACs is given in table 1. ■

Physical synthesis faces up to design challenges

Gary Meyers, Synplicity

Gary Meyers, President and CEO of Synplicity: "Current design challenges go beyond design size and complexity"

■ When I started in the FPGA business, FPGAs contained only tens of logic cells and designs were easily captured with graphical schematic capture software. Benefiting from Moore's Law trends, FPGAs grew bigger over time and by the early 1990s, FPGAs had grown to thousands of logic cells and capturing designs with schematics was becoming tedious. Device complexity and capability were outrunning the capabilities of the design entry tools. Fortunately, there was a solution waiting in the wings: logic synthesis. Logic synthesis had been commercially developed a few years before, mostly aimed at the ASIC market, and combined the efficiency of RTL abstraction with a concise text-based syntax for design description. In time, these benefits, coupled with logic synthesis engines optimized for FPGAs, caused a fundamental shift in the way FPGAs were designed. By the late 1990s, logic synthesis had nearly replaced schematic capture.

Logic synthesis has served us well. Because of this technology, today's designers routinely complete designs that consist of tens or even hundreds of thousands of logic cells. We are however starting to see some strains on the technology as it is tasked with ever larger design challenges. For example, with larger devices and longer runtimes, timing closure has become a difficult problem. The designer may know that he can reach his required speed, but getting there requires multiple time-consuming design flow iterations to converge. This condition has become common enough that various tool strategies for coping with it have reached the market, such as floorplanning or post-synthesis optimizations. But these solutions have been only partially successful in solving the timing closure problem. As it turns out, current design challenges go beyond design size and complexity. In this deep-submicron era there are new issues to be reckoned with. Power has now become a problem in many FPGA designs. Soon other issues encountered in deep-submicron SoC design such as on-chip clock variation will start to resist hardware-centric solutions or will cap performance gains well below physical device limits. FPGA vendors are doing an excellent job of handling these effects so that they are invisible to FPGA users but from here on the task is going get more difficult with each generation of technology.

Just as in the early 1990s, we now need a transition to a new design technology that can provide the basis for coping with both the increased design complexity and the threatening intrusion of deep submicron issues into the realm of digital FPGA design. What we need is a transition from logic synthesis to physical synthesis.Because physical synthesis simultaneously combines logic optimization with placement and routing, the technology has the information needed to tackle problems ranging from timing closure to power optimization. Physical synthesis provides the needed platform to tackle both rising complexity and deep submicron issues. Like with previous transitions this one is only a matter of time. ■