flexiMAC

A MICROSEQUENCER FOR FLEXIBLE PROTOCOL PROCESSING

A Lattice Semiconductor White Paper

February 2006

Lattice Semiconductor
5555 Northeast Moore Ct.
Hillsboro, Oregon 97124 USA
Telephone: (503) 268-8000
www.latticesemi.com

Introduction

FPGAs have undisputed advantages in flexibility, zero NRE costs, faster time to market and in-system programmability. Their relative disadvantages, compared to an ASIC solution, are in cost, performance and power consumption. Traditionally, ASICs have been used for high volume, low cost devices or applications in which very high performance is required.

The move to 90nm technologies has greatly increased the performance and density of FPGAs. At the same time, the soaring cost of reticule sets has made ASIC solutions uneconomic in 90nm 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 flexiMAC hard core embedded into the LatticeSCM 90nm FPGAs 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. Initially, Lattice has developed microcode for 1G Ethernet MAC, 10G Ethernet MAC, PCI Express Data Link layer and Physical layer functions for x1, x2 and x4 lane channels and Advanced Switching Interconnect (ASI) data link layer functions.

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 [1].

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 a LUT based implementation of the same functions. The area saving is greatest for 10Gbps data rates in which data-path functions such as CRC checking need a 64bit 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.

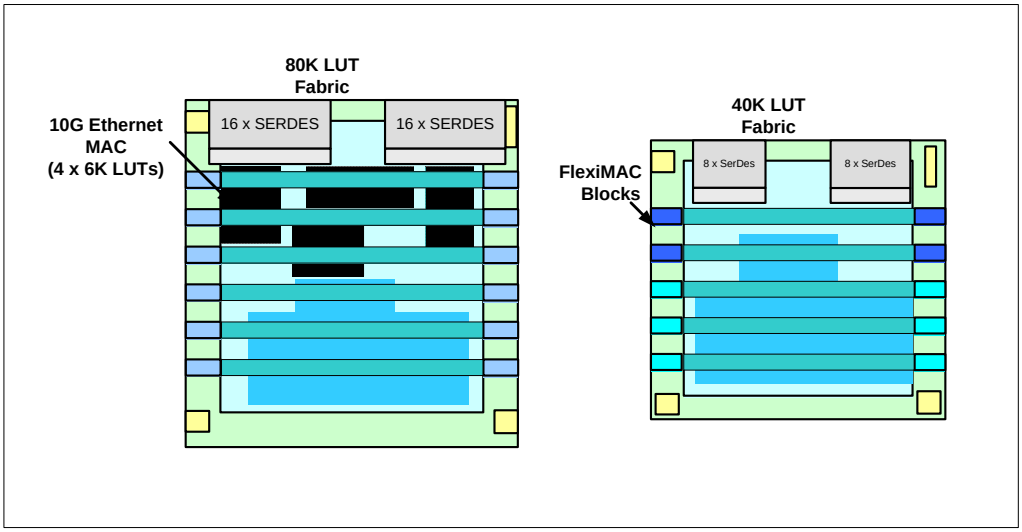


Figure 1 - 10G Ethernet Cost Reduction via flexiMAC

Figure 1 shows how the flexiMAC can be used to reduce system cost. In this case a 4 port 10Gbit Ethernet switch is ported from a 60K 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.

flexiMAC Architecture

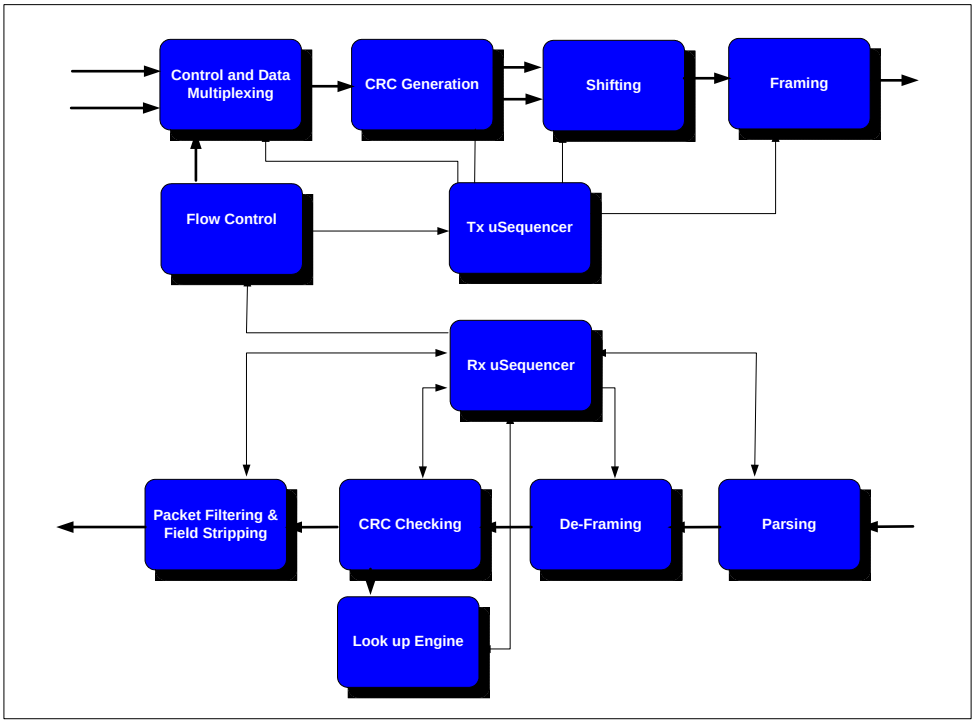
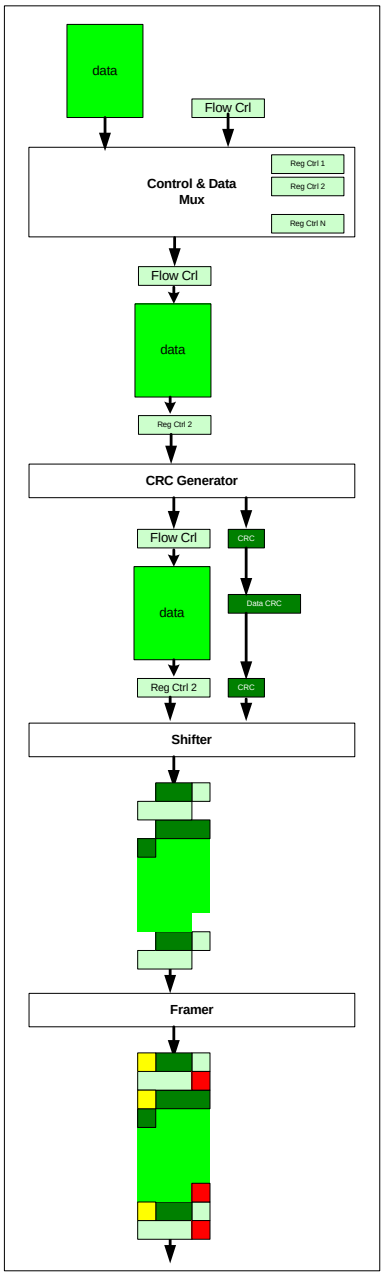


Figure 2 - flexiMAC Architecture

Figure 2 shows the architecture of the flexiMAC packet processor. The flexiMAC processes both the transmit and the 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 the receive sides are controlled by a microcoded microsequencer that is described in more detail later in this white paper. The following two sections describe the behaviour of the transmit and receive paths of the flexiMAC.



Transmit Path

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, Ethernet PAUSE frames can be constructed in this block or PCI Express Initialization packets defined. The source of the transmitted packets is controlled by the microsequencer.

Packets from the Control & 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 function of the transmitter is captured pictorially in Figure 3.

Figure 3 -Transmit Framing

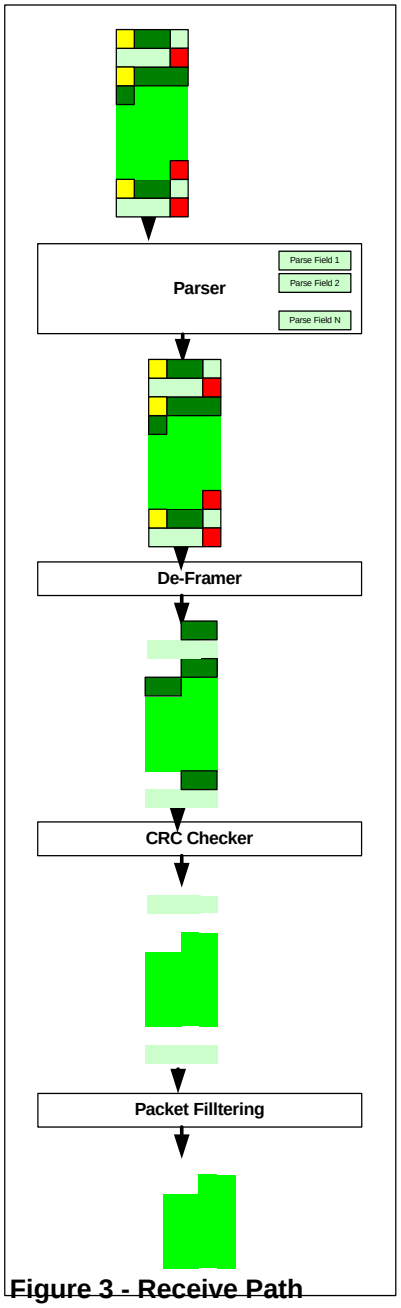


Figure 3 - Receive Path

Receive Path

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 realigns 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.

The function of the transmitter is captured pictorially in Figure 4.

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.

Fleixible Microsequencer Architecture

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 10Gbps. 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 pre-amble 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.

Using the flexiMAC in FPGA Designs

The initial release of LatticeSC FPGAs will have multiple integrated flexiMACs to support serial protocols running over the embedded SERDES. Initially there will be microcode releases for 1G Ethernet, 10G Ethernet MAC and x1, x2 and x4 PCI Express or Advanced Switching Interface (ASI) support.
The following table gives the savings in LUTs over a soft IP version mapped to the FPGA fabric.

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

Ethernet Implementation

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.

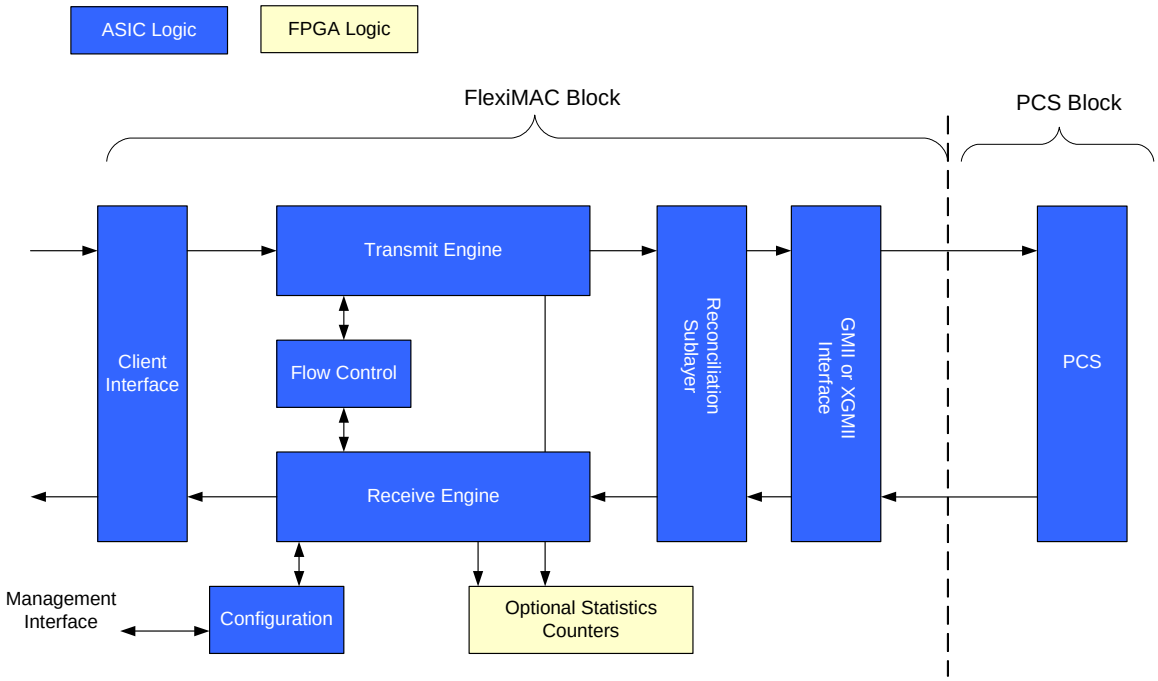


Figure 4 - Ethernet MAC + PCS Partitioning

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.

PCI Express Implementation

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.

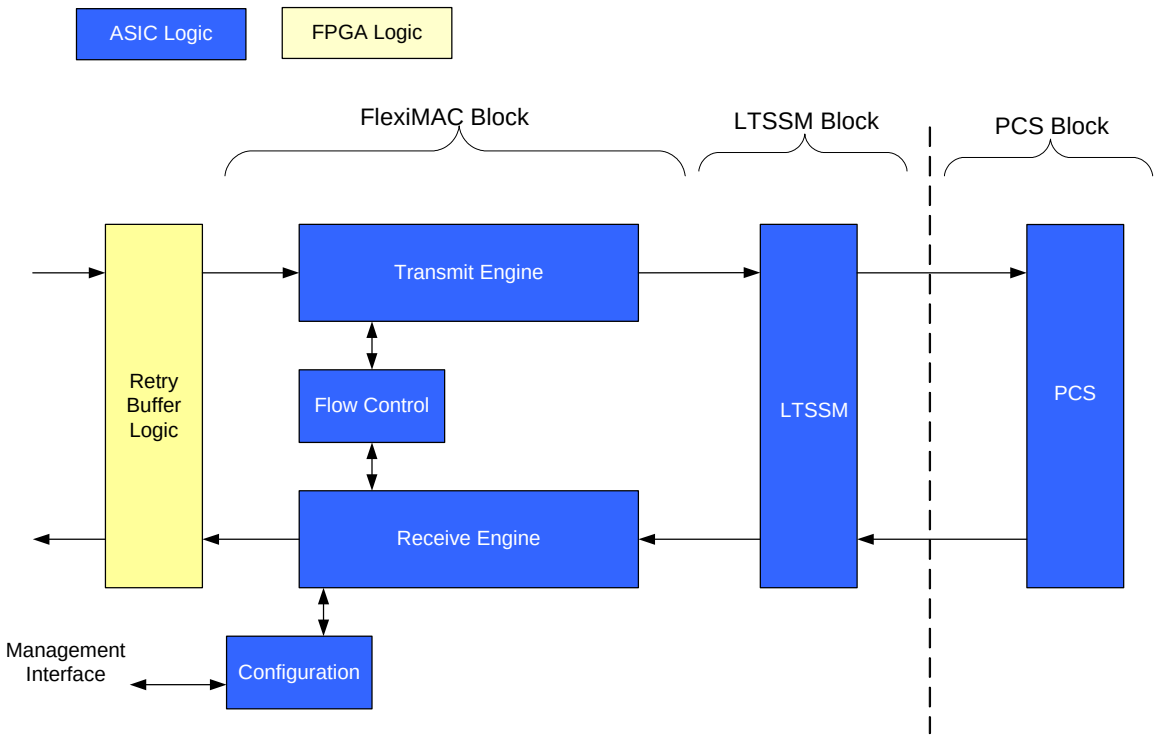


Figure 5 – PCI Express Data Link Layer + PCS Partitioning

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
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
5. Lane alignment

Invoking a PCI Express node in 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 resource saving over a soft IP solution for PCI Express, as shown in Table 1.

IspLEVER Support

Integrating the FlexiMAC into designs is simple using Lattice ispLEVER design tools. FlexiMAC configurations can be selected using a simple GUI interface, and test benches and simulation models can be selected to ease integration into customer designs, saving design time, LUT usage and power.

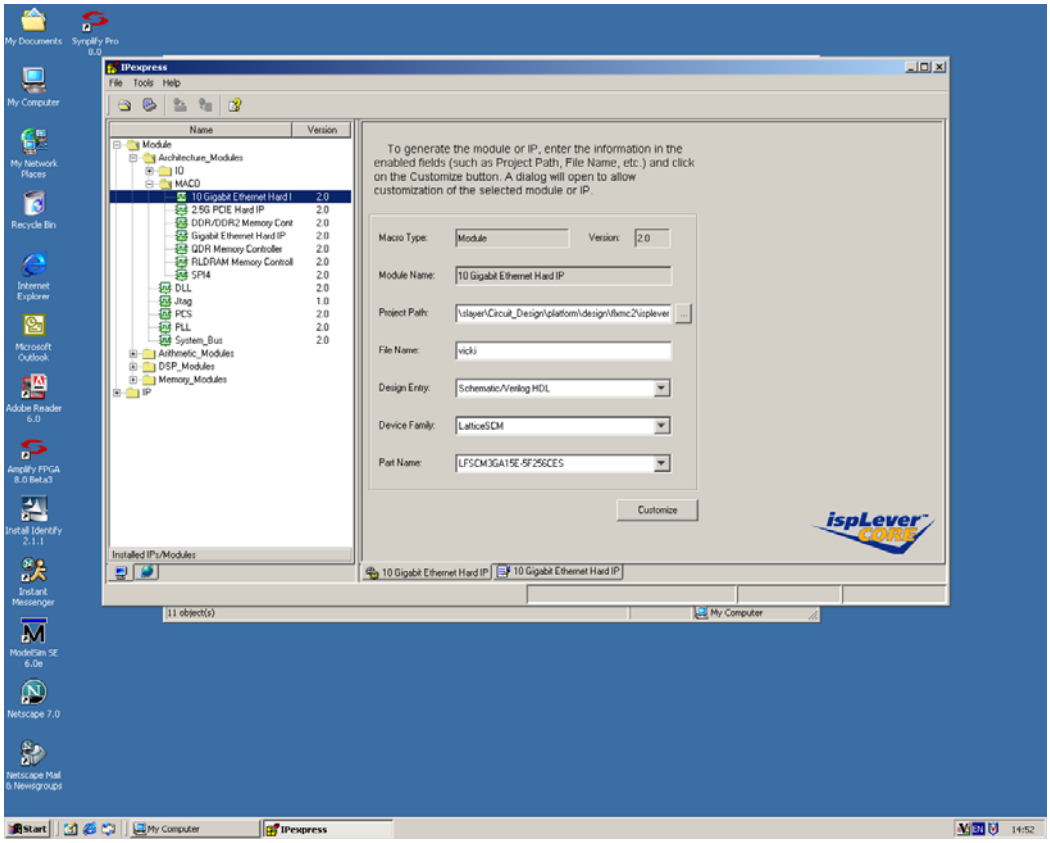


Figure 6 - Instantiating a flexiMAC 10GE MAC via ispLEVER

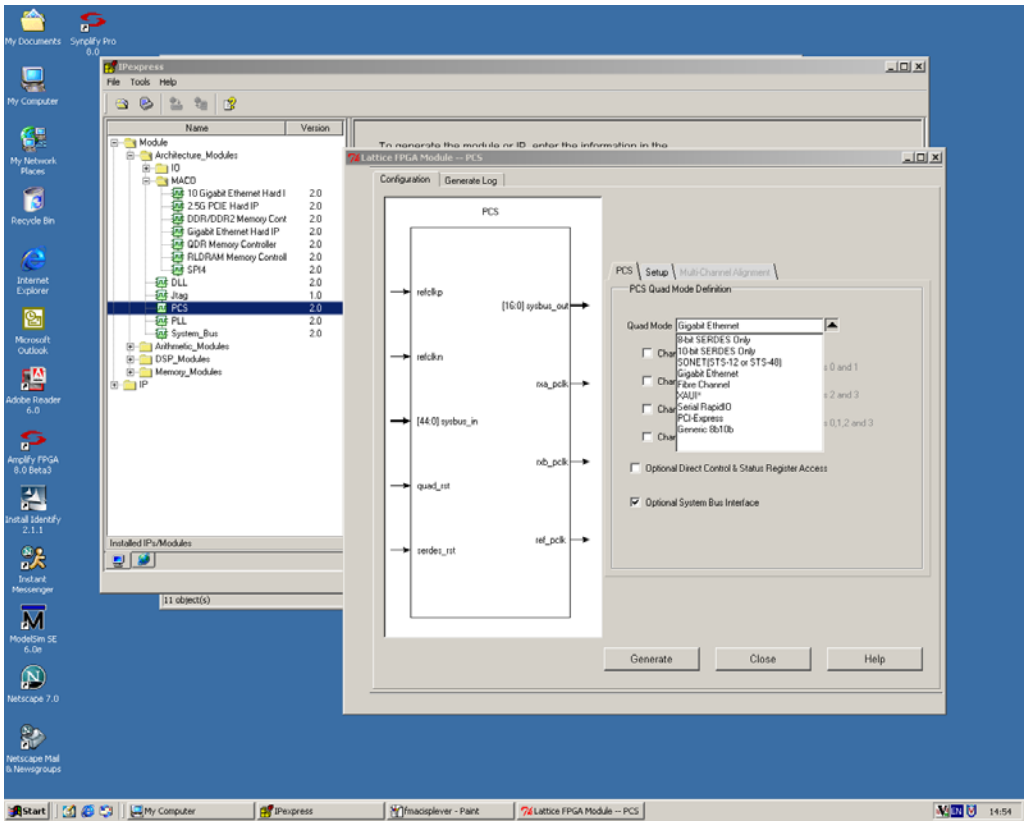


Figure 7 - Selecting PCS Mode in ispLEVER

Conclusion

When selecting FPGAs to implement high speed Serial protocols such as 10G Ethernet or PCI Express, LatticeSCM products have hard core devices to support both physical layer functions in the programmable Physical Coding Sublayer PCS (see [1]) and data link layer functions in the flexiMAC, making them the most efficient programmable solution available today for these protocols.

Integrating the PCS or flexiMAC into designs is easy using the IspLEVER design tool and for designs targeting these ubiquitous protocols provides a significant saving in FPGA logic, power and time to market.