A Scalable Fiber Optic Local Area Network Demonstrator

Albert Kar-Hing Au

Department of Electrical and Computer Engineering
McGill University, Montréal, Canada.

June, 2000.

A thesis submitted to the Faculty of Graduate Studies and Research, in partial fulfillment of the requirements for the degree of Master of Engineering.

Abstract

The continuing advances in silicon integrated circuit technology in accordance with Moore's Law have fueled the constant performance improvement of computing systems. At the same time, the deployment of networks of workstations with high-speed processors has driven the need to increase bandwidth capacities in networked environments. Electrical interconnect technologies are plagued by substantial physical problems at high frequencies, that limit their data transmission capacities. With increasing clock rates, high-bandwidth computing and communication architectures become more difficult to implement using electrical interconnects, and therefore creates a strong motivation for exploring both free-space and fiber-based optical interconnect technologies.

In this thesis, the development of a Fiber Optic Local Area Network (LAN) demonstrator is described. The demonstrator will be used as a testbed for research in high-speed networking technologies, lean protocols, and bandwidth-intensive network-oriented applications, which began at McGill University, Canada, and is being continued at McMaster University, Canada.

A complete LAN system would consist of an array of 16 Personal Computers (PC), where each PC has a Network Interface Card (NIC), with a parallel fiber optic datalink to a centralized optoelectronic switch core. The centralized switch core can support up to 102.4 Gbit/s of aggregate data traffic, generated by the 16 NICs. The architecture of the demonstrator is designed to scale to Terabits of bandwidth using optoelectronic integrated circuit technologies, i.e., integrated Complementary Metal-Oxide Semiconductor (CMOS) substrates with Vertical Cavity Surface Emitting Laser (VCSEL) optical transmitters and Self Electro-optic Effect Device (SEED) modulator optical receivers.

A subset of the complete system has been constructed and is operational. A prototype NIC card using the Motorola Optobus VCSEL transceivers for the optical datalinks has been constructed and is described. A prototype high-speed bipolar switch core, using statically configurable electrical positive emitter coupled logic (PECL) 16x16 crossbar switches and Motorola Optobus transceivers has been constructed and is described. We have successfully demonstrated the transmission of high-speed packetized data from one NIC card, through 10 meters of parallel fiber ribbon and the centralized switch core, and back to the NIC. This thesis will summarize our experiences on the

design and testing of our first demonstration system, and the development towards a Terabit switch core. A paper summarizing this demonstrator has been published in 2000 in the journal Applied Optics: Information Processing.

Résumé

Le développement continuel des technologies de circuits intégrés au silicium, qui plus est, de manière exponentielle, a entraîné le développement constant, des systèmes informatiques. Cependant, l'émergence et le déploiement à large échelle de stations de travails informatiques bâties autour de puissants microprocesseurs a exacerber le besoin d'augmenter de manière significative les capacités en terme de largeur de bande des réseaux qui ont en résultés. Les interconnections électriques ont montrés leur limites au niveaux physique lorcequ'il s'agit de transmettre des données en haute fréquence. De ce fait, l'implémentation des nouvelles architectures de communication à large bande et des systèmes de traitement de l'information à cadence d'horloge élevée devient ardue, d'où la motivation d'explorer d'autres technologies, telles que celles basées sur l'interconnection optique.

Cette thèse, décries le développement d'un démonstrateur de réseau local (LAN) à fibre optique. Le démonstrateur, sera utilisé comme base de test dans la recherche dans les technologies de réseaux à haute vitesse, des protocoles ainsi que différentes applications dédiées aux réseaux requérant large bande.

Un système LAN complet, serait constitué d'un ensemble de 16 ordinateurs personnels (PC), où chaque PC possède une Carte d'Interface Réseaux (NIC), relié à un commutateur optoélectronique central par le biais d'une liaison parallèle à fibre optique. Le commutateur central peut supporter un agrégat, en terme de trafic de données, de 102.4 Gbits/s générés par les 16 NICs. L'architecture du démonstrateur est tel, que la largeur de bande est de l'ordre du Terabit. Cette architecture utilise les technologies de circuits intégrés optoélectroniques, c.à.d, substrats (CMOS) avec transmetteurs optique à laser à émission de surface à cavité vertical (VCSEL) et récepteur optique à dispositif à effet auto electro-optique (SEED).

Une partie du système complet a été construite et présentement opérationnelle. Un prototype de la carte NIC utilisant l'émetteur-récepteur VCSEL Optobus de Motorola pour les liaisons optiques a été également réalisé. Un prototype de commutateur bipolaire a haute vitesse, utilisant des commutateurs 16x16 crossbar et des émetteurs-récepteurs Optobus de Motorola à logique PECL statiquement configurable, a été réalisé et est décris dans cette thèse. Nous avons, avec succès, démontré la transmission à haute vitesses de

paquets de données d'une carte NIC à travers 10 mètres de ruban de fibre parallèle et à travers un commutateur central, avant que ces données ne refassent le chemin inverse vers la carte. Cette thèse résumera nos expérience concernant la conception et le test de notre premier système démonstrateur et le développement de commutateurs à vitesses de l'ordre du Terabits.

Acknowledgments

I would like to begin by thanking my thesis supervisor, Prof. Ted Szymanski, for his guidance and support, and for introducing me to such an exciting field of study. I wish him continued success at McMaster University, Canada.

I wish to thank the Research Computing Systems Management support staff for keeping the MACS Lab running smoothly. Access to mechanical tooling equipment was provided by the workshop staff in the Department of Electrical & Computer Engineering and the School of Architecture. Thanks to Emmanuelle Laprise for her help with testing our initial Optobus samples.

Many past and present students and professors in the MACS Lab have enriched my academic and non-academic experience. It has been a great pleasure to meet and to work with my colleagues, Boonchuay Supmonchai, Thomas Obenaus, Victor Tyan, James Wong and Myriam Lafrenière-Roula. I thank them for their friendship and help throughout my graduate degree. Special thanks to Mourad Oulmane for translating the abstract.

Finally, and most importantly, I would like to thank Queenie Kwok and my family for their continual support, understanding and encouragement.

Funding Acknowledgments

The research presented in this thesis was funded in part by Natural Sciences and Engineering Research Council (NSERC) of Canada Individual Research Grant (IRG) held by Prof. T. H. Szymanski (number OGP0111211601), and in part by a grant from the former SPAR Space Systems.

This research was carried out in the Microelectronics and Computer Systems (MACS) Laboratory at McGill University. Computing equipment, computer aided design tools and device fabrication were supplied through the Canadian Microelectronics Corporation (CMC).

Contents

List of Figures

List of Tables

CHAPTER 1

Introduction

The continuing advances in Complementary Metal-Oxide Semiconductor (CMOS) technology in accordance with Moore's Law have enabled the development of high-performance processors and highly integrated systems-on-a-chip. Over the past two decades, the performance of computing systems has been increasing by roughly a factor of 10 every 5 years. Inexpensive Personal Computers (PCs) have been constantly improving in performance, and commercialization of processors with 1 GHz clock rates has commenced [1][2]. Significant computing resources can be achieved at reasonable cost by using clusters of such PCs in a high-bandwidth networked environment.

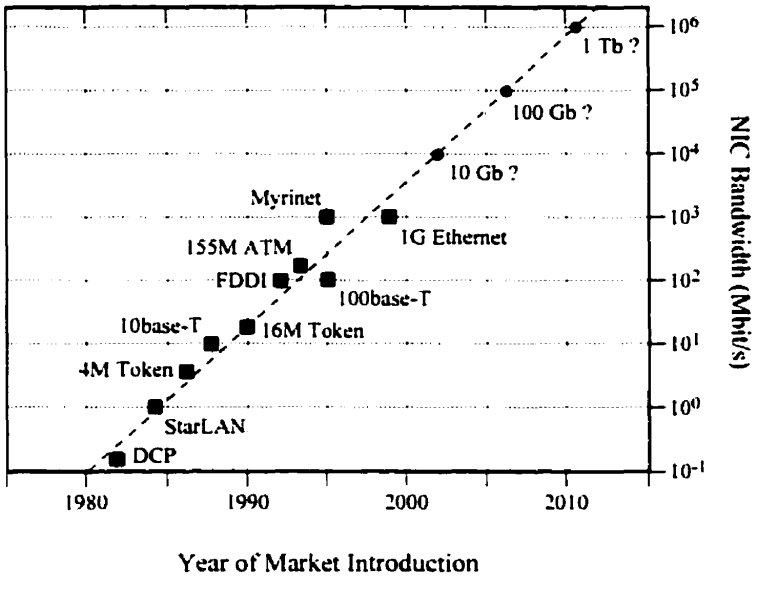


Fig. 1.1. Evolution of Network Bandwidth Measured at the Network Interface Card (NIC) [3].

Driven by Networks of Workstations [4], the performance of Local Area Network (LAN) systems has been increasing at the rate of roughly a factor of 10 every 4 years, as shown in Fig. 1.1. For example, Ethernet technology has progressed from 1 Mbit/s (StarLAN) in the early 1980s, to 10 Mbit/s (10Base-T) in the late 1980s, to 100 Mbit/s (100Base-T) in the early 1990s, and now to 1 Gbit/s in the late 1990s [5][6]. 10 Gbit/s

Ethernet is now being standardized [7]. Recently, the transmission of Ethernet at 10 Gbit/s has been demonstrated on a development optical network system based on Dense Wavelength Division Multiplexing (DWDM) [8].

All projections indicate that the trend will continue for the next 2 decades. Hence, 100 Gbit/s LAN technology is expected within 10 years, as shown in Fig. 1.1. According to [9], 100 Terabit interconnect technology is expected in high performance electronic systems by the year 2004, such as the 100 TeraFlop computing systems being planned by the United States Department of Energy's Accelerated Strategic Computing Initiative (ASCI) project. With the rapid growth in network capacity, it is difficult to predict what new bandwidth-intensive applications may evolve over the next decade. However, traditional applications such as scientific computing will always demand very high bandwidth networking [9], and new applications such as teleconferencing, virtual reality, graphics visualization and distributed computing will likely accelerate the need for high bandwidth communication networks [6].

1.1. Limits of Electrical Interconnects

Electrical interconnection technologies currently dominate the communication of information because they are often the most inexpensive approach and are known to be reliable. However, electrical interconnections have substantial problems at high frequencies, which limit the transmission capacity of the interconnect.

Copper cables are bulky and restricted in density, bandwidth, and interconnect distance capabilities. At long distances and high bit rates, electrical interconnects suffer from signal attenuation and distortion due to the unavoidable skin effect in practical cables. The skin effect [10] is the physical phenomenon that, at high frequencies, current is carried only in a thin layer near the surface of the conductors. Electrical wires behave as antennas, both for transmitting and receiving, and leads to electromagnetic interference (EMI) and frequency-dependent crosstalk. Other physical problems include skew, impedance matching, wave reflection phenomenon, power dissipation and interconnect density limitations [11].

The idea of an empirical "upper bound" to the bit rate capacity B which can flow in a simple digital electrical interconnection was proposed in [12]. This limit is defined by the *aspect ratio* of the interconnect $l/\sqrt{A}$, where l is the length and A is the total cross-sectional area of the wiring, and is approx. $B \approx B_0 A/l^2$ bit/s. The constant $B_0 \approx 10^{15}$ bit/s for high-performance cables and printed circuit board traces, and $B_0 \approx 10^{16}$ bit/s for on-chip VLSI (very-large-scale-integration) traces. In practice, the physical problems

mentioned above will set in well below this predicted upper limit, but nevertheless, the aspect ratio limit provides a "rule of thumb" boundary at which the wire can no longer be regarded as reliably transmitting the logic voltage from one end to the other.

With increasing clock rates, high-bandwidth architectures such as switching fabrics and multiprocessing systems become more difficult to implement with electrical interconnections, and therefore creates a strong motivation for exploring optical interconnect technologies. Improving the performance of electrical interconnects is an alternative approach to address the network bandwidth bottleneck, but at the expense of increased delay, complexity, power dissipation and cost. Recent attempts at enhancing the bandwidth of electrical interconnects are equalized serial line [13], asymmetric serial link [14] and simultaneous bi-directional signaling [15]. The equalized serial line scheme uses transmitter pre-distortion equalization circuitry to compensate for the frequency dependent loss in the wire over the operating frequency range. This technique can provide 4 Gbit/s of bandwidth over 6 m of twisted pair wires [13]. The asymmetric serial line scheme transmits data on both edges of its internal clock, and uses 2 receivers to recover alternating data bits. This scheme can achieve bandwidths of 2 Gbit/s per wire [14]. In the simultaneous bi-directional signaling scheme, signals are transmitted simultaneously in both directions using multi-level signaling. Bandwidths of 2.5 Gbit/s per wire at several centimeters on a printed circuit board has been demonstrated [15].

1.2. Need for Optical Interconnects

Optics solves or relieves most of the physical limitations of electrical interconnections. Optical fiber-based interconnections have received considerable attention due to the potential systems level advantages offered by optics. Parallel fiber ribbons constructed from multimode fibers are a low cost solution to provide the needed interconnect bandwidth. Designers of large computer systems are attracted by the advantages of size, essentially unlimited bandwidth, immunity from electromagnetic interference (EMI), higher edge connection density and the performance at distances not achievable by traditional copper links.

Current parallel optical datalinks feature ribbon cables with 10 or more channels and support data transfer rates up to 2.5 Gbit/s per channel at distances of a few hundred meters, as shown in Table 1.1. Such high-bandwidth communication capability cannot be realized using electrical interconnects.

Project	**OETC** [16]	**Optobus** [17]	**Jitney** [18]	**PAROLI** [19]	**POLO-2** [20]	**PONI** [20]
Participants	GE, IBM, AT&T, Honeywell	Motorola	IBM, 3M, Lexmark	Siemens, Infineon	HP, AMP, DuPont, SDL, USC	Agilent, USC
No. of Channels (Tx / Rx)	32 / 32	10 / 10	20 / 20	12 / 12	10 / 10	12 / 12
Fiber Pitch (μm)	140	250	500	250	250	250
Wavelength (nm)	850	850	850	840	850	850
Data Rate per Channel (Gbit/s)	0.625	0.80	0.50	1.25	1.00	2.5
Aggregate Bandwidth (Gbit/s)	20.0	8.0	10.0	15.0	10.0	30.0
Max. Link Distance (m)	100	300	100	300	300	100
Optical Power per Channel (mW)	0.70	0.80	2.0	0.25	1.0	1.0
Electrical Power Consumption (W)	6.3 (Tx) 2.0 (Rx)	1.6	2.3 (Tx) 3.7 (Rx)	1.2 (Tx) 0.8 (Rx)	2.5	1.2 (Tx) 1.2 (Rx)
Electrical I/O	LVDS, ECL	ECL, CML	LVDS	LVDS	ECL	LVDS

OETC is Optoelectronic Technology Consortium; PAROLI is Parallel Optical Link; POLO is Parallel Optical Link Organization; PONI is Parallel Optics for Network Interconnects; Tx is Transmitter chip; Rx is Receiver chip; LVDS is Low Voltage Differential Signaling; ECL is Emitter Coupled Logic; CML is Current Mode Logic.

Table 1.1. Comparison of Current Parallel Optical Datalinks.

1.3. Fiber Optic Local Area Network

This thesis describes the development of a Fiber Optic LAN demonstrator [6][21]. The demonstrator will be used as a testbed for research in high speed networking technologies, lean protocols, and bandwidth-intensive network-oriented applications. The basic system architecture of our LAN demonstrator is described in detail in [5][22], and the demonstrator is part of a longer term effort to develop a Scalable Terabit Optical LAN.

A complete system consists of an array of 16 Pentium-based PCs, running the Linux operating system. Each PC would have a *Network Interface Card* (NIC), with a parallel fiber optic datalink to a *centralized optoelectronic switch core*, supporting approx. 6.4 Gbit/s per NIC. The centralized switch core processes the data generated by 16 NICs, for an aggregate bandwidth of up to 102.4 Gbit/s. With LAN capacity increasing by a factor of 10 times every product generation (3-4 years), this current demonstrator is

roughly 2 product generations ahead of current Gigabit Ethernet LAN technology. However, the demonstrator is designed to scale to Terabits of bandwidth capacity by using emerging optoelectronic technologies, such as integrated CMOS substrates with Self Electro-optic Effect Device (SEED) modulator optical I/O [23], or integrated CMOS substrates with Vertical Cavity Surface Emitting Laser (VCSEL) optical transmitters [24].

A subset of the complete demonstrator has been constructed. A prototype NIC has been developed using high-speed bipolar electrical serial-to-parallel converters and CMOS field programmable gate arrays (FPGAs). A prototype optoelectronic Switch Core Printed Circuit Board (PCB) has been developed using statically configurable electrical PECL 16x16 crossbar switches and CMOS FPGAs. Motorola Optobus transceivers [17] are used for the optical datalinks between the prototype NIC and the prototype optoelectronic Switch Core PCB. Transmission of high-speed data over the LAN, from one NIC card, through 10 meters of parallel fiber ribbon and the centralized optoelectronic switch core, and back to the NIC is demonstrated.

As in typical networks, the communication protocols are an integral element of the Scalable Terabit LAN architecture. The demonstrator described in this thesis highlights the hardware integration of optical and electrical components into a functional system. Protocol issues due to the interaction between protocol layers will be investigated in subsequent work in this project.

1.4. Author's Contribution

This research, under the guidance of Prof. T. H. Szymanski, is part of a long term project to develop a scalable terabit optical network, and is financed through NSERC IRG Grant OGP0112116O1. The overall architecture is described in [5], and elaborated upon in [6], [21], [22], [25], [26], [27], and [28]. The author contributed to the conceptual design, and had direct responsibility for the design, implementation and testing of the demonstrator components and system. A paper describing this research has been published in the journal Applied Optics: Information Processing [21]. Some supporting development work was performed in undergraduate student projects supervised by Prof. T. H. Szymanski [29][30][31].

1.5. Thesis Organization

This thesis focuses on the system prototypes constructed for a subset of the LAN, and is organized as follows. Chapter 2 presents an overview of the system architecture of the scalable terabit optical LAN, and describes the key components of the network.

These components include the parallel fiber technology, the workstation NIC and the centralized optoelectronic 16x16 switch core. The detailed design of the NIC is provided in Chapter 3. The NIC performs serialization and deserialization functions on the datapath between the workstation's electronic system bus and the optical fibers. The NIC contains a Protocol Processor, implemented in programmable hardware, to manage the LAN communication protocols. The detailed design of the optoelectronic Switch Core PCB is provided in Chapter 4. The complete 16x16 optoelectronic Switch Core PCB performs the switching functions using electrical PECL crossbar switches, which are configured using a programmable hardware-based Switch Controller. The Routing Software used to control and to configure the switch core is also presented. Prototypes of the NIC and the Switch Core PCB are constructed for the LAN demonstrator, and their implementation details are given in Chapter 5. Experimental results from the transmission of high speed data over the LAN are presented in Chapter 6, and illustrate the operation of the prototypes. The switch core scales smoothly from electronic switch ICs with discrete optical transceivers to a single-chip optoelectronic IC solution. This transition to a switch core IC with integrated optical I/O is discussed. Finally, Chapter 7 draws the conclusion of the work and suggests some future work that could arise from this research.

CHAPTER 2

Network Architecture

An overview of the Fiber Optic Local Area Network (LAN) architecture is presented. The LAN is designed to support general inter-computer communication and emphasizes high-bandwidth optoelectronic switching rather than optical switching. The key components of the network are described in Section 2.2. These components are the parallel fiber technology, the Network Interface Card (NIC) on the workstations and the centralized 16x16 optoelectronic switch core. Related optical networking projects of interest are discussed in Section 2.3.

2.1. Communication Networks

A communication network consists of nodes interconnected by communication links in some topology. A node can be broadly defined as an information processing unit (e.g., a computer workstation, a processor in a multiprocessing system, a shared data storage facility). The nodes interact by exchanging messages across the network through their network interfaces using standardized protocols. A message contains long strings of binary bits and is assembled into shorter fixed size segments called packets for transmission. The communication links form the transmission medium of the network, and the links may be electrical or optical. In conventional networks, messages are carried by electrical signals over copper cables. In optical networks, messages are transmitted using optical beams, which propagate within fibers or over free-space.

Communication networks are classified, primarily based on the network topology, into these 4 categories [32]: shared-medium networks, direct networks, indirect networks, and hybrid networks. The Fiber Optic LAN described in this thesis is an *indirect* or *switch-based* network, in which the nodes are indirectly connected using switches [32]. The ports of a switch are connected to nodes or to the ports of other switches. The interconnection of these switches defines the network topology, and examples are crossbar networks, multi-stage interconnection networks, and fat-trees.

An $M \times N$ switch is a device with M input links and N output links (usually $M = N$), and is shown in Fig. 2.1. An input link is a communication channel for a switch to receive messages from other nodes or switches. An output link is a communication

channel for a switch to transmit messages to other nodes or switches. The function of the switch is to route messages from an input link to an output link.

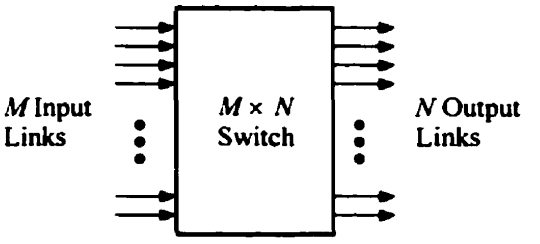


Fig. 2.1. Graphical Representation of a $M \times N$ Switch.

2.2. Fiber Optic LAN Architecture

The architecture of the Fiber Optic LAN is shown in Fig. 2.2, and is described in detail in [5][22]. The network interconnects 16 PC workstations to a centralized switch core, and can be called an "optical star" based upon its centralized star-like topology. Each workstation in Fig. 2.2 has a dedicated parallel fiber ribbon connection from its Network Interface Card (NIC) to the 16x16 optoelectronic switch core, and provides a bandwidth of 6.4 Gbit/s per link.

The architecture explores and unifies several novel approaches to achieve multi-gigabit networking [5]. Low cost *parallel fiber technology* is used to link the workstations to a centralized optoelectronic switch core. Parallel fiber technology can support 10's of fibers and 100's of gigabits of bandwidth, and can simplify many fundamental network design decisions. The network supports *lean protocols* implemented in programmable hardware [5]. Lightweight protocols can reduce the interaction overhead between the workstation Central Processing Unit (CPU) and the network interface, and can decrease the end-to-end communication latency, thus resulting in more efficient movement of network data [33]. Programmable hardware will allow the network to support the latest and most efficient protocols.

A departure from conventional design approach is the *de-emphasis of buffering* within the switch core [5]. Buffer memory and controllers occupy the majority of an integrated switch core. By pushing the buffers back into the workstations, where memory is abundant, the complexity and the VLSI area of the switch core can be reduced. The switch core employs very fast *one-pass arbitration schemes* for packet routing [25]. With *extensive pipelining*, this reduced complexity switch core can operate at very high internal clock rates [25]. At the system level, given the low skew (<1 ps/m) and high bandwidth in

optical fibers, it is attractive to *broadcast timing*, synchronization, and control from the centralized switch core. This approach can eliminate synchronization circuits and deskew buffers used to maintain data alignment, and can allow simpler communication protocols.

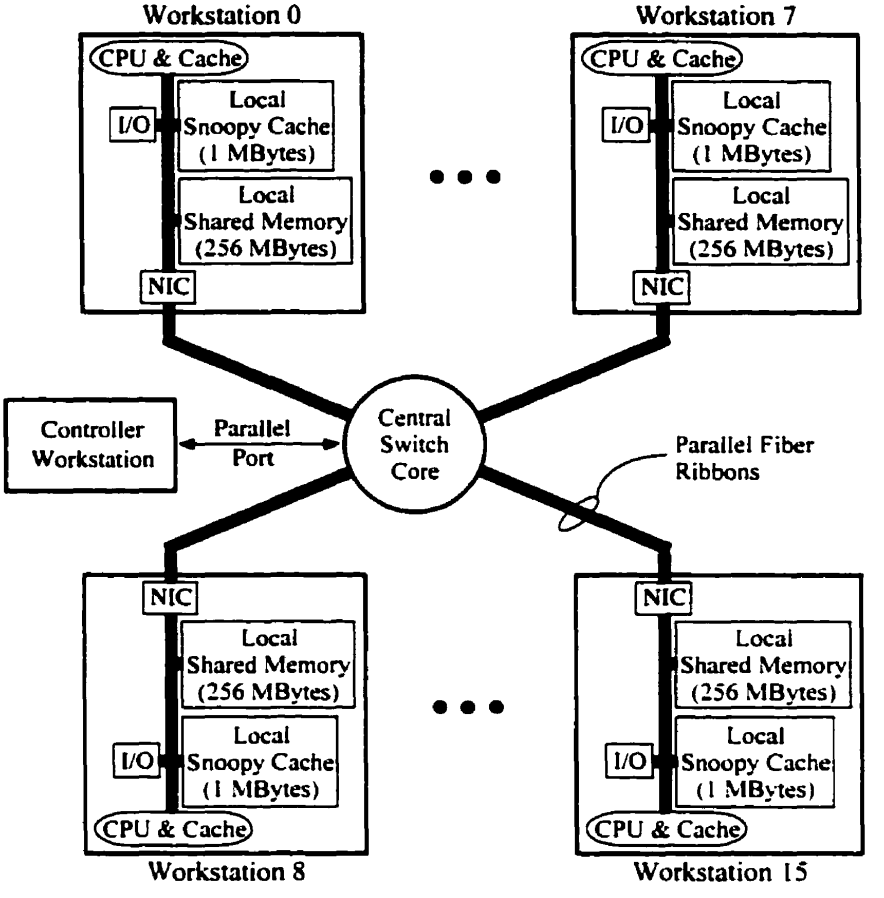


Fig. 2.2. Architecture of the Fiber Optic Local Area Network.

2.2.1. Parallel Fiber Optic Datalinks

Motorola Optobus [34] fiber optic datalinks are used for the optical interconnects between the workstations and the switch core. From a physical layer functional viewpoint, the Optobus transceiver is a passive electrical-to-optical (E/O) and optical-to-electrical (O/E) signal converter. Each Optobus datalink consists of two 10-channel multimode parallel fiber ribbons and supports optical clock rates of 800 MHz per channel. The Optobus is a discontinued commercial product, and this transceiver was used in our prototypes to demonstrate proof-of-concept. Alternative optical transceiver technology for parallel fiber ribbons, such as PAROLI [19] and POLO-2 [20], can be used

to implement the optical links. A detailed description of the Optobus transceiver is given in Section 3.2.4.

2.2.2. Optoelectronic Switch Core

The functional design of the centralized switch core is shown in the functional diagram of Fig. 2.3. The switch core processes the data generated by the 16 NICs, for an aggregate bandwidth of 102.4 Gbit/s. The parallel fiber ribbons from the 16 workstations terminate into optical receivers, which form the inputs to the 16x16 switch, where each input and output port is 8 bits wide. The switch contains up to 8 crossbar switching elements, with each one operating on a bit-slice of the input datapath. The 16 switch output ports interface to optical transmitters, which propagate the packets to their destinations over the outgoing parallel fiber ribbons. There are 2 ways to build the switch core design illustrated in Fig. 2.3, and they are discussed as follows.

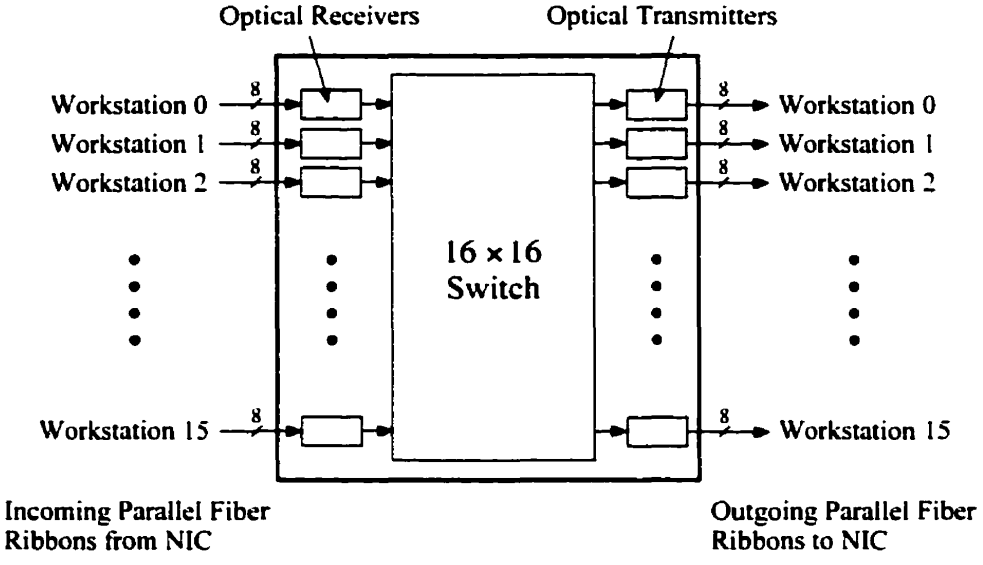


Fig. 2.3. Optoelectronic Switch Core Functional Diagram.

The first approach is termed the *optoelectronic integrated circuit (OEIC) implementation*. The centralized switch core can be implemented using a single-chip OEIC, which incorporates optical and electronic devices on a single substrate. The switching circuits are fabricated on a high-density CMOS VLSI die, which is then integrated with opticai receivers and VCSEL optical transmitters [24], resulting in a single-chip optoelectronic switching IC. The optical signals would be coupled directly between the parallel fiber ribbons and the optoelectronic IC through an imaging system

[5]. This single-chip optoelectronic switch core can potentially scale to terabit per second (Tbit/s) bandwidth [5] with progressing optoelectronic technologies and growing multimode fiber transmission capacities [35].

The second approach is termed the *discrete component implementation*. The switching function in the switch core of Fig. 2.3 can be performed electrically using parallel ICs (e.g., CMOS Field Programmable Gate Arrays [5], commercial switch chips [36]) or custom switching ASICs [25]. The parallel fiber ribbons terminate into discrete optical transceiver modules, which perform O/E and E/O conversions on the received and transmitted messages at the switch I/O ports. Such an "optoelectronic" switch core, interconnecting 16 workstations, can be implemented on a single high speed PCB to support an aggregate bandwidth of 102.4 Gbit/s [5]. The design and prototyping of such an optoelectronic switch core PCB is the focus of this thesis.

2.2.3. Network Interface Card

The optical datalink supports clock rates of 800 MHz, while the electronic CPU memory bus within the workstation operates typically at 100 MHz. The NIC interfaces between these two domains of different clock rates, and resides directly on the high bandwidth memory bus rather than the low bandwidth I/O bus. The functional diagram of the NIC is shown in Fig. 2.4.

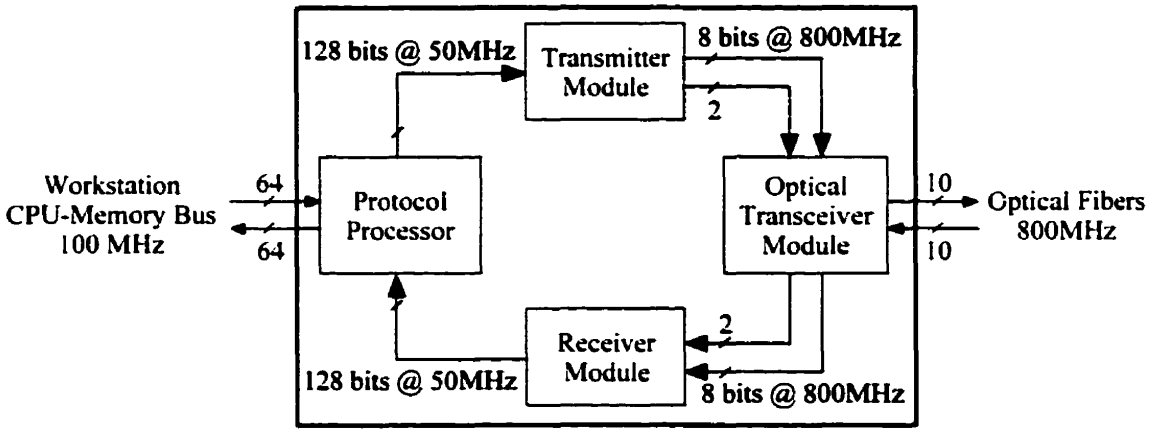


Fig. 2.4. Network Interface Card Functional Diagram.

The NIC offloads end-to-end network functions from the workstation CPU, and consists of 4 modules: the Protocol-Processor, the Transmitter Module, the Receiver Module and the Optical Transceiver Module (OTM), which is based on the Motorola Optobus. The *Protocol Processor* implements the communication protocols required for

the LAN in Field Programmable Gate Array (FPGA) hardware [5]. The protocols are lean and spans 2 traditional network functions: Data Link Control (DLC) and Automatic Repeat Request (ARQ). The DLC protocol [37] performs error-detection and optional error-correction over a single Optobus datalink. The hardware-based ARQ protocol [37] performs end-to-end functions, such as sliding window flow control, message-fragmentation and reassembly.

In Fig. 2.4, the NIC *transmit datapath* extends from the Protocol Processor electrical outputs to the OTM optical outputs. The NIC *receive datapath* extends from the OTM optical inputs to the Protocol Processor electrical inputs. The Protocol Processor provides a 128-bit wide CMOS datapath for data to be transmitted. The *Transmitter Module* serializes the 50 MHz 128-bit CMOS datapath, and generates an 8-bit PECL datapath clocked at 800 MHz to be transmitted by the OTM over the parallel fiber ribbon. The *Receiver Module* deserializes the 800 MHz 8-bit PECL datapath from the OTM, and generates a 50 MHz 128-bit CMOS datapath, which is fed to the Protocol Processor for processing.

In order to provide a mechanism for accurate timing recovery, block coding schemes, such as 8B/10B, can be used on the serial byte datapath. These schemes allow for the encoding of control symbols and timing information to provide data frame alignment and synchronization. However, given the low skew and low delay in optical fibers, it should be feasible to operate the entire network "synchronously". As described in [5], 2 channels in each fiber ribbon are reserved for broadcasting of timing, synchronization and control from the switch core. One channel transmits the *bit* clock, i.e., the 800 MHz clock used by the Receiver Module. The second channel carries the *frame* signal used to denote the start of a packet frame. The remaining 8 channels support bytes of data, at the 800 MHz clock rate. These signals allow for a "self-timed" design, where each receiver uses the bit clock of the sender to sample the data. This approach simplifies synchronization within the network, since all workstations receive timing information from the low skew fibers, which are all 10 meters long in our localized network.

2.3. Literature Review

This section reviews several projects featuring relevant network and switching architecture attributes.

Myrinet [38] is a switched LAN technology developed by Myricom. Variable length packets are wormhole-routed through the network of 1.28 Gbit/s links and crossbar switches. The NIC at each node connects to the workstation I/O bus (typically, 32 bits

wide at 33 MHz) and contains a programmable network interface processor. The Beowulf project [39] originated from the earth and space sciences computing community, and harnesses the parallel processing power of clustered PCs, interconnected using mass-market commodity network components (e.g., switched Ethernet, Myrinet).

In our discussion, optoelectronic switch core designs are categorized into single-chip OEIC or discrete component implementations. Examples of the first category include the following. The AMOEBA switch [3] is a 12.8 Gbit/s 16x16 optoelectronic crossbar switch, implemented using integrated CMOS-SEED technology. Each switch port supports 50 MHz 16-bit datapaths. Each 16-bit datapath is wavelength multiplexed onto a single fiber of the 1x32 fiber array. A 16 Gbit/s 64x64 optoelectronic crossbar switch, implemented by integrating InGaAs semiconductor on silicon CMOS is described in [40]. Each switch port supports a serial datapath at 250 MHz. Each of the 64 optical inputs are fanned out 64 times, and the set of 4096 optical beams are relayed onto optical receivers on the OEIC. A 8 Gbit/s 16x16 optoelectronic 3-stage Clos crossbar switch, implemented using integrated CMOS-SEED technology is described in [41]. Each switch port supports a serial datapath at 500 MHz. Each stage of the Clos network resides on a different OEIC, and the stages are interconnected optically in free-space.

Examples of the second category include the following. In [42], the design of a 160 Gbit/s 64x64 Asynchronous Transfer Mode (ATM) switching system is described. Each port has a 32-bit datapath and can be multiplexed onto an optical link at the OC-48 rate (2.4 Gbit/s). The switching system contains twenty-four 8x8 CMOS switch element ICs arranged in a 3-stage Benes network. In [43], the design of a 128 Gbit/s 32x32 optoelectronic distributed ATM switch for the iPOINT (Illinois Pulsar-based Optical Interconnect) testbed is proposed. Each port has a 4-bit datapath clocked at 1 GHz and interfaces to optical fibers through Metal-Semiconductor-Metal (MSM) photodetectors and GaAs double-quantum-well lasers. The proposed switch core considers a GaAs multi-chip module (MCM) implementation, while the prototype switch core utilizes CMOS FPGAs and 1300 nm wavelength optical transceivers.

2.4. Summary

The Fiber Optic LAN architecture has been reviewed. Several novel attributes of the architecture are, (i) the use of low cost parallel fiber technology, (ii) the use of lean protocols in programmable hardware, (iii) the de-emphasis of buffering within the switch core, (iv) one-pass arbitration schemes for packet routing, (v) deep pipelining in the

switch core, and (vi) the broadcasting of timing and synchronization information throughout the LAN [5].

In principle, this LAN architecture, using off-the-shelf technology, can supply interconnect requirements of up to a few Tbit/s by exploiting parallel switch cores. Each LAN could switch approx. 100 Gbit/s of data, and several switch cores could be interconnected with fiber ribbons using various standard topologies, such as 3-stage Clos networks or Fat-Trees, and can be used as a building block for multi-Terabit optical LANs.

The Fiber Optic LAN connects 16 workstations to a centralized 16x16 optoelectronic switch core. Motorola Optobus datalinks are used for the optical interconnects between the workstation NICs and the switch core. The NIC implements the network functions in programmable hardware, and provides a 800 MHz 8-bit datapath to the switch core. The switch core processes the data generated by the 16 NICs, for an aggregate bandwidth of 102.4 Gbit/s. The switch core can be implemented on a high speed PCB containing discrete parallel switching ICs and optical transceiver ICs.

The complete designs of the NIC and the optoelectronic switch core are given in Chapters 3 and 4.

CHAPTER 3

Network Interface Card Design

This chapter presents the design of a Network Interface Card (NIC), which can deliver a bandwidth of 6.4 Gbit/s to and from the Switch Core Printed Circuit Board (PCB). Section 3.1 begins with an overview of Positive Emitter Coupled Logic (PECL), and describes the basic PECL logic gate structure and its operation in both single-ended and differential signaling modes. High-speed termination techniques, which are applied throughout the prototypes, are discussed. Readers familiar with these topics should proceed to Section 3.2. The NIC architecture is outlined in Section 3.2. Each NIC contains the Protocol Processor, the Transmitter Module, the Optical Transceiver Module (OTM), and the Receiver Module. The electronic components used to build these modules are discussed.

3.1. Emitter Coupled Logic (ECL) at 5 Volts

Emitter Coupled Logic (ECL) is a form of non-saturating bipolar logic based on bipolar junction transistors (BJTs) [44]. The ECL logic gate contains an input differential amplifier to amplify the signal, and output emitter followers to logically combine the input signals. The BJTs in the logic gate operate within their active regions, and can change states rapidly. Hence ECL circuits are widely used in high-speed applications (i.e., above several hundred MHz).

Traditionally, ECL requires negative power supplies ($V_{CC} = 0V$, $V_{EE} = -5.2V$). Positive Emitter Coupled Logic (PECL) is simply ECL operated from positive supplies ($V_{CC} = 5V$, $V_{EE} = 0V$). PECL specifies a logic low (0) voltage $V_L = 3.3$ V and a logic high (1) voltage $V_H = 4.1$ V. The switching reference voltage is mid-way in between at $V_{BB} = 3.7$ V [45]. Fig. 3.1 shows the basic PECL gate with single-ended input V_{IN} and differential outputs V_{OUT} and $\overline{V_{OUT}}$. The PECL gate consists of 3 subcircuits: the *differential amplifier*, the *bias network*, and the *emitter followers*, as highlighted in Fig. 3.1.

The bipolar junction transistors (BJTs) Q1 and Q2 form the differential amplifier input stage. The PECL input V_{IN} is connected to the base junction of Q1, while the base junction of Q2 is biased at the reference voltage V_{BB}, which is generated by the bias network. As shown in Fig. 3.1, the differential amplifier outputs (V_{C1} and V_{C2}) at the

collector junctions of Q1 and Q2 drive the base junctions of the emitter followers Q3 and Q4, which serve as voltage level shifters and low impedance output drivers. Typically, the input stage supplies (V_{CC1}) and the output stage supplies (V_{CC2}) are isolated, as shown in Fig. 3.1. This scheme reduces the coupling of output voltage transients onto the differential amplifier and the bias network, thereby minimizing noise within the latter subcircuits.

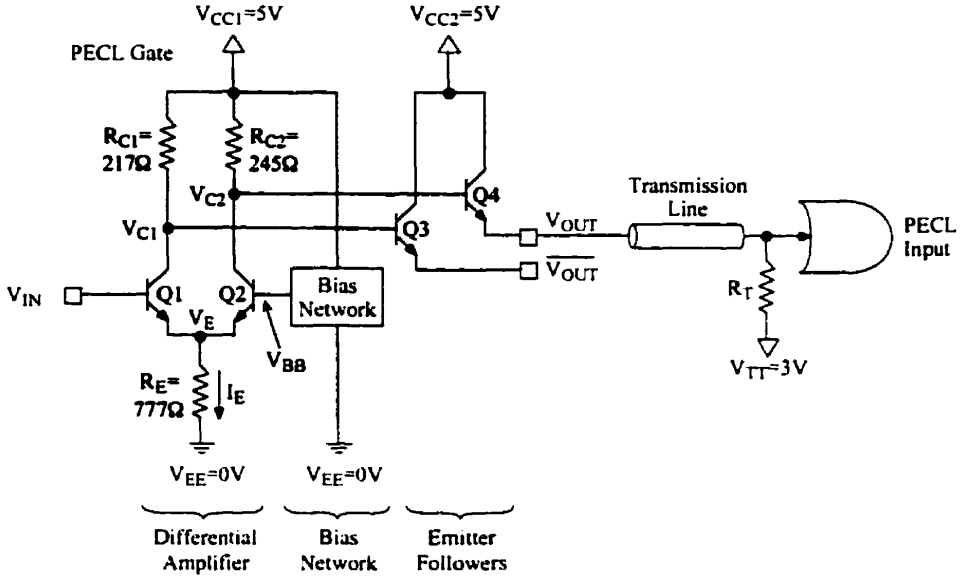


Fig. 3.1. Basic Positive Emitter Coupled Logic Gate Circuit (Adopted from [44]).

The operation of the PECL gate in Fig. 3.1 is as follows. When input $V_{IN} = V_H = 4.1V$, Q1 is on (i.e., forward-biased) and Q2 is off. For a forward-biased BJT, the base-emitter junction voltage drop is $V_{BE} = 0.8V$. Referring to Fig. 3.1, the common emitter voltage $V_E = V_{IN} - V_{BE} = 3.3V$ and the emitter current through resistor R_E is $I_E = (V_E - V_{EE})/R_E = 4.25mA$. The current flowing through resistor R_{C1} is approx. equal to I_E [44]. Hence, the differential amplifier outputs $V_{C1} = V_{CC1} - I_E R_{C1} = 4.08V$, and $V_{C2} = 5V$ since no current flows through resistor R_{C2}. At the output stage, V_{C1} and V_{C2} are shifted down by $V_{BE} = 0.8V$ because Q3 and Q4 are forward-biased. Therefore, the output voltages are $V_{OUT} = 4.20V$ and $\overline{V_{OUT}} = 3.28V$.

When input $V_{IN} = V_L = 3.3V$, Q1 is off and Q2 is on (i.e., forward-biased). Referring to Fig. 3.1, the common emitter voltage $V_E = V_{BB} - V_{BE} = 2.9V$ and the emitter current through R_E is $I_E = (V_E - V_{EE})/R_E = 3.73mA$. The current flowing through R_{C2} is approx. equal to I_E [44]. Hence, the differential amplifier outputs $V_{C2} = V_{CC1} - I_E R_{C2} = 4.09V$, and $V_{C1} = 5V$ since no current flows through R_{C1}. At the output stage, V_{C1} and V_{C2} are shifted

down by $V_{BE} = 0.8V$ because Q3 and Q4 are forward-biased. Therefore, the output voltages $V_{OUT} = 3.29V$ and $\overline{V_{OUT}} = 4.20V$.

The emitter followers of Q3 and Q4 have low output impedance (under 10 Ω), and can drive terminated transmission lines in single-ended or differential modes. Fig. 3.1 shows the output V_{OUT} driving an external transmission line, which is terminated at the receiving PECL input to a terminating voltage V_{TT} through a resistance R_T. The terminating resistor also functions as a pull-down resistor for the driver's emitter follower, which sources current in both the logic high and low states. Termination techniques are further described in Section 3.1.2.

The logic levels (V_L and V_H) are temperature dependent, due to the temperature variation of the base-emitter junction voltage drop (V_{BE}) in a forward-biased BJT. The bias network in Fig. 3.1 generates a reference voltage V_{BB}, which tracks with temperature in a predetermined manner [44]. The bias network ensures that V_{BB} is centered mid-way between V_L and V_H to guarantee noise immunity across the operating temperature range.

3.1.1. Electrical Signaling Schemes

Single-ended PECL signaling uses one wire to transfer a binary signal (i.e., logic 1 or 0) from the driver to the receiver. The transmitted voltage signal V_{IN} is detected at the receiver by comparing with the local reference voltage V_{BB}, as shown in the input stage of Fig. 3.1.

In differential PECL signaling, a binary signal is sent over a pair of wires by driving one wire with the signal and the other wire with the signal's complement. Referring to Fig. 3.1, a PECL gate with differential inputs is realized by replacing the reference voltage V_{BB} with the input signal's complement $\overline{V_{IN}}$ connected to the base of Q2. The receiver measures the differential voltage $\Delta V = V_{IN} - \overline{V_{IN}}$ between the wires to detect the binary signal. The differential voltage swing must be larger than the minimum input sensitivity of the receiver.

Differential signals provide better noise immunity than single-ended signals [45]. Firstly, the effective voltage swing ($V_H - V_L$) across the two wires is twice that of single-ended signaling ($V_H - V_{BB} = V_{BB} - V_L$). Secondly, differential lines are unaffected by variations in logic levels caused by power supply fluctuations and operating temperature, since the receiver senses the differential voltage ΔV rather than detecting absolute logic levels. Shifts in logic and reference levels in single-ended lines will directly reduce the noise margin. Thirdly, any noise will appear equally on both wires, so ΔV is unaffected. As a result, the receiver will not detect noise as long as the differential voltage is within the common mode range (CMR) of the receiver. For single-ended lines, noise generated on the

signal line by crosstalk and inductive coupling directly reduces the noise margin.

3.1.2. High-Speed Signal Termination

The use of high-speed PECL logic in our prototypes required a great deal of attention to the analog effects of high-speed signal propagation. Fig. 3.2 shows a PECL logic gate (A) driving a transmission line (wire or PCB trace) connected to another PECL logic gate (B). The transmission line has characteristic impedance $Z_o = 50\Omega$. The capacitance C_L represents the capacitive load of the receiver.

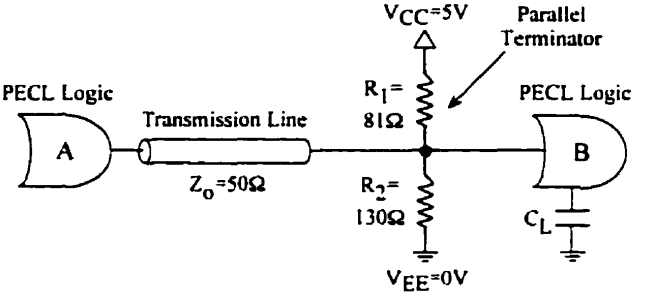


Fig. 3.2. Thevenin equivalent Parallel Termination.

The response of a wire to an incoming high-speed signal depends on its physical length and the rise time of the signals passing through it. The effective length of a signal rising edge L can be computed as:

$$L = \frac{T_r}{D} \qquad \text{(Eq. 3.1)}$$

where L is the effective length of a signal rising edge (inches)
T_r is the signal rise time (ps)
D is the wire propagation delay (ps/inch)

Termination is required when the physical wire length driven by a logic gate is more than $L/6$ [10], because the wire is considered as a transmission line and the response of the incoming signal is distributed along the wire. Without proper termination, reflections occur at both ends of the wire and interfere with the original waveform. The PECL components in the prototypes have a rise time of 1000 ps. If we assume a wire propagation delay of 160 ps/inch [10], the effective length L calculated from Eq. 3.1 is approx. 6.3 in. and all signal wires longer than approx. 1 in. must be terminated.

Terminators may be placed at the driver or at the receiver [10] and two common

techniques using passive components are "series" and "parallel" termination. In *series termination*, a terminating resistor is placed between the driver and the transmission line. Let R_{ST} denote the series terminating resistance and R_O denote the output resistance of the driver. The sum of the series terminating resistance and the driver output resistance must equal the characteristic impedance of the transmission line, i.e., $R_{ST} + R_O = Z_O$. Due to the voltage divider at the driver end, the signal propagates at *half amplitude* down the line, reflects at the receiving end, and is absorbed by the series terminating resistor at the driver. The reflected signal (half amplitude) at the receiver is superimposed on the original signal (half amplitude), causing a full amplitude waveform to appear at the receiver.

In *parallel termination*, the transmission line is terminated at the receiver to a termination voltage V_{TT} (3 V for PECL) through a resistor $R_T = Z_O$. Typically, the driver output resistance is much lower than the line impedance, and the signal waveform propagates at *full amplitude* down the line. All reflections are absorbed by the terminating resistor, which should be placed beyond the last receiver. Loads (receivers) can be distributed anywhere on a parallel terminated transmission line because the full logic swing is available along the line. A variation of the parallel termination, called *Thevenin equivalent parallel* or *split termination*, does not require an additional termination voltage, and is employed in our prototypes. Such a termination scheme is highlighted in Fig. 3.2. The Thevenin equivalent of the resistor divider R_1 and R_2 is equal to R_T terminated to V_{TT}. The values of R_1 and R_2 can be calculated from Eqs. 3.2 and 3.3 [46]. For PECL applications, $V_{CC} = 5\text{V}$, $V_{EE} = 0\text{V}$, $V_{TT} = 3\text{V}$ and $Z_O = 50\,\Omega$, resulting in $R_1 = 83\Omega$ and $R_2 = 125\Omega$.

$$R_2 = \left(\frac{V_{CC} - V_{EE}}{V_{CC} - V_{TT}}\right) Z_O \qquad \text{(Eq. 3.2)}$$

$$R_1 = \left(\frac{V_{CC} - V_{TT}}{V_{TT} - V_{EE}}\right) R_2 = \left(\frac{V_{CC} - V_{EE}}{V_{TT} - V_{EE}}\right) Z_O \qquad \text{(Eq. 3.3)}$$

where V_{CC} is the positive supply voltage (V)
V_{EE} is the negative supply voltage (V)
V_{TT} is the termination voltage (V)
Z_O is the characteristic impedance (Ω)

In our prototypes, commercially available SIP (single in-line) terminating resistors with $R_1 = 81\Omega$, $R_2 = 130\Omega$ and ±5 % tolerance are used. Fig. 3.3 shows a 10-pin SIP package containing 8 Thevenin equivalent parallel terminators (i.e., pairs of R_1, R_2). Pins 1 and 10 are connected to 0 V and 5 V, respectively. Pins 2 through 8 terminate the signal transmission lines. Despite its simplicity, a drawback of this termination scheme is the high

static power dissipation. Each terminator in Fig. 3.3 consumes approx. 0.25 W, which is roughly 5 times more than the basic parallel termination scheme (50 Ω into 3 V).

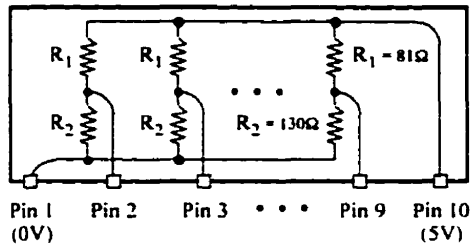


Fig. 3.3. Internal Schematic of Single In-line Package Terminating Resistors.

3.1.3. Notation

This section describes the naming and labeling conventions used throughout the remainder of this thesis.

- Input-output port names and signal names of components and entities are written in uppercase, e.g., serial data output DOUT.
- Active low signals are denoted with the lowercase prefix "n", e.g., control signal nSTROBE.
- Differential signaling requires 2 signal lines. The complementary signal line of a differential signal is denoted with a horizontal bar above the name, e.g., clock input STRBIN and $\overline{\text{STRBIN}}$.
- Vectors or multi-bit signals are denoted by appending the range of the bit numbers after the name, e.g., 4-bit input address INADDR0..3.

3.2. Design of the Network Interface Card

The NIC within the workstations can deliver 6.4 Gbit/s of bandwidth to the Switch Core PCB. As shown in Fig. 3.4, the complete NIC consists of the Protocol Processor, the Transmitter Module, the Receiver Module, and the Optical Transceiver Module (OTM). The *NIC transmit datapath* extends from the Protocol Processor electrical outputs to the OTM optical outputs. The *NIC receive datapath* extends from the OTM optical inputs to the Protocol Processor electrical inputs. The Transmitter Module uses 20% of the available link bandwidth to implement Conditional Invert with Master Transition (CIMT) line encoding, which is described in Section 3.2.2.1. The 4 modules are constructed using off-

the-shelf electronic components, and are described in detail in Sections 3.2.1 through 3.2.4. The descriptions contain detailed technical design information regarding the functionality and operation of the IC components. Readers who wish a broad overview may skip the following:

- Subsections 3.2.2.1 to 3.2.2.4 in the discussion of the Transmitter Module.
- Subsections 3.2.3.1 to 3.2.3.4 in the discussion of the Receiver Module.
- Subsections 3.2.4.1 and 3.2.4.2 in the discussion of the OTM.

These sections are denoted by the asterisk symbol *.

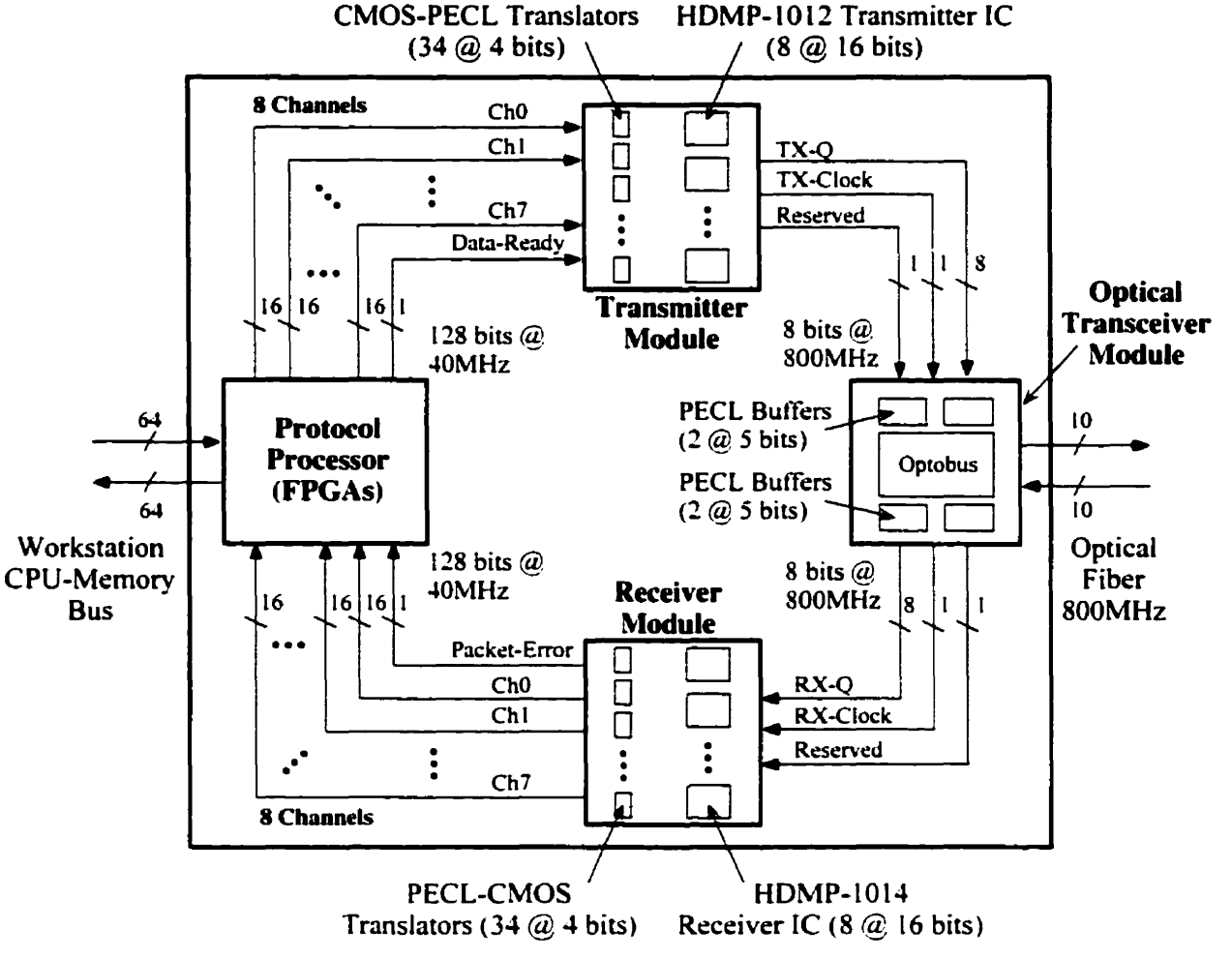


Fig. 3.4. Design of the Network Interface Card.

3.2.1. Protocol Processor

The Protocol Processor in Fig. 3.4 provides the 40 MHz frame clock, the 128-bit parallel datapaths and the necessary control signals for the Transmitter and Receiver Modules, and is implemented using CMOS Altera FLEX 81500 FPGAs [47]. The FPGA device has a 280-pin PGA (pin grid array) package, with 208 user I/O, 60 power-ground pins and 12 dedicated configuration I/O. A complete Protocol Processor utilizes approx. 400 I/O and requires at least 2 FPGAs. An architectural overview of the Altera FLEX FPGA is provided in Appendix A.

The I/O signals of the Transmitter Module, the Receiver Module and the Optical Transceiver Module in the NIC are compatible to the PECL voltage levels, where a PECL logic low is 3.3 V and a PECL logic high is 4.1 V. Conversion between CMOS (logic low is 0 V and logic high is 5 V) and PECL signals are performed for the datapaths and control signals between the Protocol Processor and the Transmitter and Receiver Modules.

3.2.2. Transmitter Module

The Transmitter Module within the NIC in Fig. 3.4 multiplexes the 128-bit datapath at 40 MHz from the Protocol Processor to an 8-bit datapath at 800 MHz. The complete Transmitter Module in Fig. 3.5 contains 8 Hewlett-Packard HDMP-1012 Gigabit transmitter ICs [48][49] and 34 Motorola 10H351 translators [50], which convert from CMOS (0-5V) to PECL (3.3-4.1V). Each transmitter IC has a 80-pin 0.80 mm lead pitch rectangular QFP (quad-flat pack) package and can serialize 16 bits at 40 MHz to generate a CIMT encoded serial stream at 800 MHz. The CIMT encoding scheme is described in Section 3.2.2.1. Each CMOS-PECL translator IC has a 20-pin DIP (dual in-line) package and operates on 4 single-ended CMOS inputs to generate 4 differential PECL outputs.

The serialization of the 128-bit datapath to 8 bits requires 8 transmitter ICs, as shown in Fig. 3.5. Each transmitter IC serializes 16 bits of the 128-bit parallel datapath from the Protocol Processor to form 1 of the 8 bit-serial data streams. Each transmitter IC requires 4 CMOS-PECL translators to convert its 16-bit data input, for a total of 32 translators. Additional translators are used to convert the transmitter IC control signal inputs from the Protocol Processor. Hence, the complete Transmitter Module requires 8 transmitter ICs and 34 CMOS-PECL translators.

The transmitter IC [48][49] within the Transmitter Module contains circuitry for frame assembly, parallel-to-serial conversion and clock generation, as shown in Fig. 3.6. The transmitter IC electrical I/O are summarized in Appendix A.

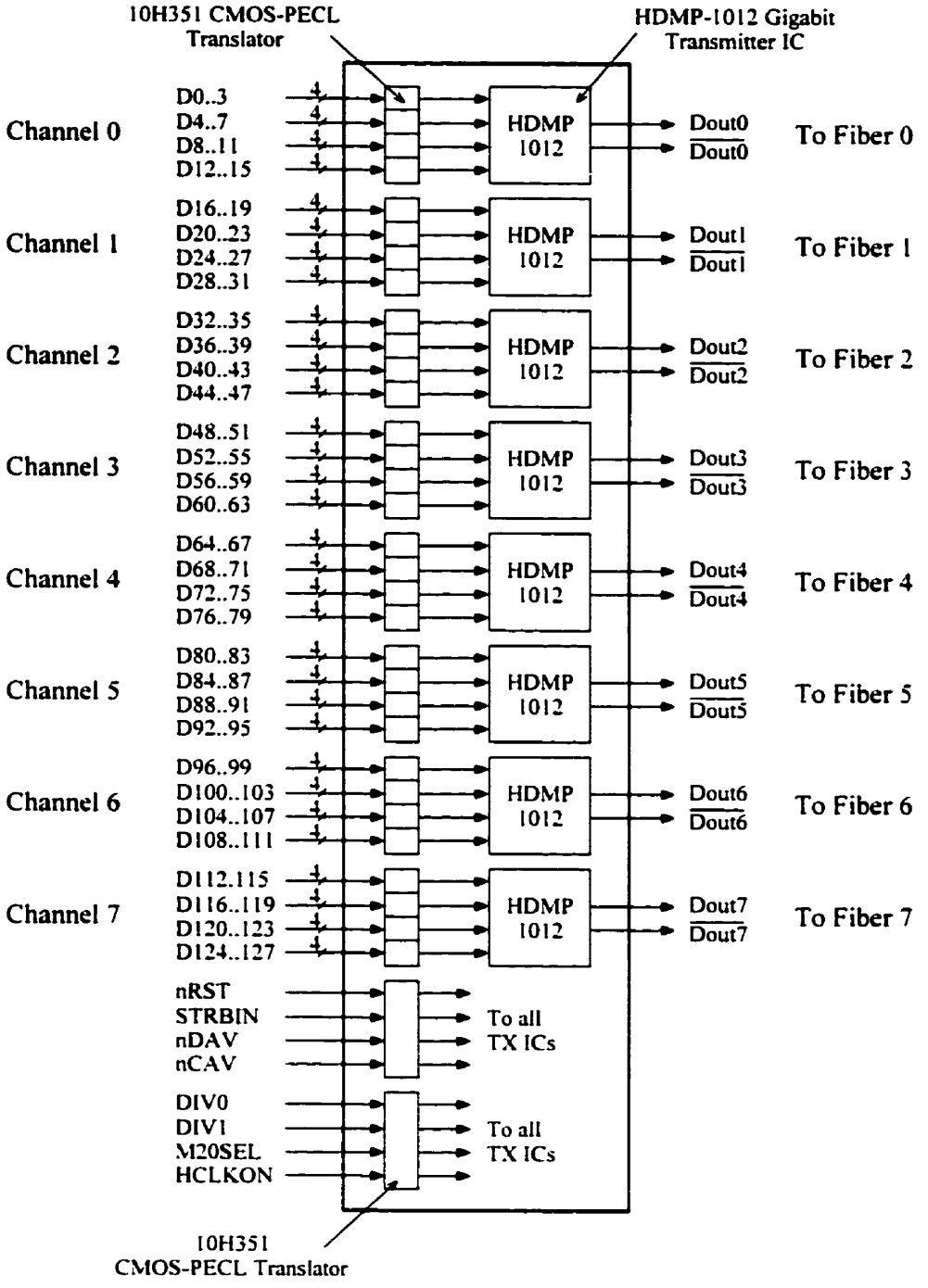


Fig. 3.5. Complete Transmitter Module.

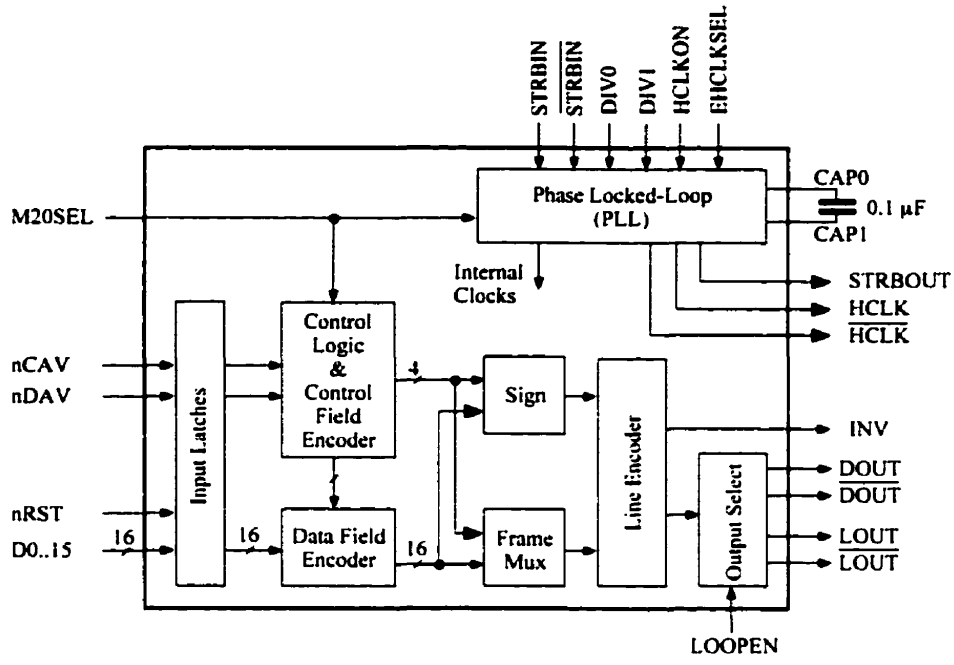


Fig. 3.6. HDMP-1012 Gigabit Transmitter IC Block Diagram.

There are 4 types of electrical I/O on the transmitter IC and the receiver IC (see Section 3.2.3). Low speed I/O are single-ended PECL. High speed input signals employ Hewlett-Packard's proprietary H50 interface [48][49], and high speed output signals employ the Buffered Line Logic (BLL) interface [48][49]. These I/O interfaces are described as follows. Readers who wish a broad overview may proceed to Section 3.2.3.

The Single-Ended PECL Input Stage [49] in Fig. 3.7 contains a differential amplifier formed by transistors Q1 and Q2. One input of the differential amplifier connects to the single-ended PECL signal while the other input is biased at the reference voltage $V_{BB} = 3.7$ V. The outputs of the differential amplifier drive internal logic circuits. The internal 16 kΩ pull-down resistor to $V_{PD} = 3.1$ V biases an unconnected input pin to logic 0. An input pin pulled to $V_{CC} = 5$ V is sensed as logic 1. The emitter follower Q3 in Fig. 3.7 forms the Single-Ended PECL Output Stage [49]. The output stage supplies a small trickle current (45 µA) which turns on the output emitter follower Q3 such that DC logic levels (i.e., static 0 or 1) appear at the output pin without external pull-down terminations. These input and output structures are consistent with those in Fig. 3.1.

The Differential H50 Input Stage [49] in Fig. 3.8 is a proprietary high speed input interface used on the transmitter and receiver ICs. The H50 input stage has integrated 50 Ω resistors built into the differential input lines. The DC level for the inputs is 5 V and all signals into H50 input pins should be AC-coupled with 0.1 µF capacitors.

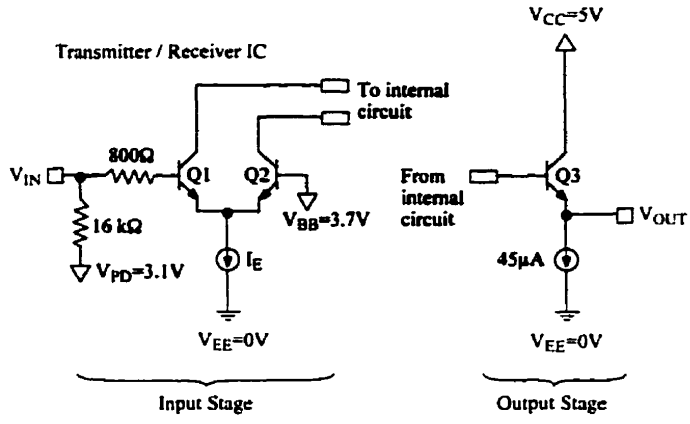


Fig. 3.7. Single-Ended PECL Input and Output Stages.

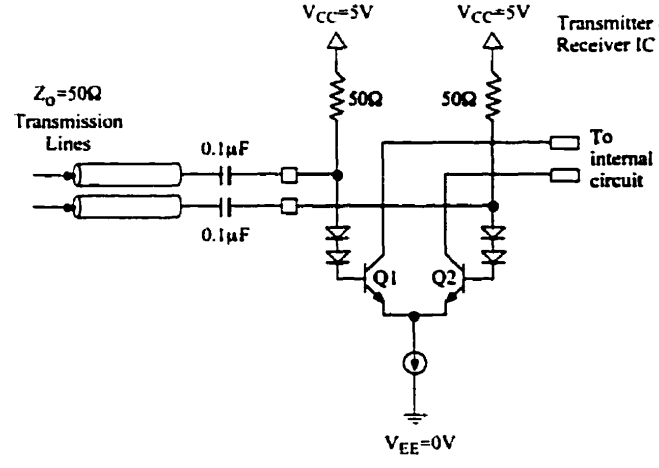


Fig. 3.8. Differential H50 Input Stage.

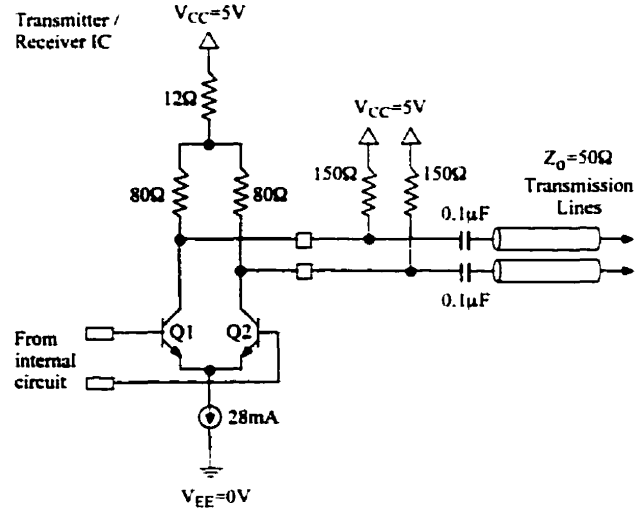


Fig. 3.9. Differential Buffered Line Logic Output Stage.

High speed outputs from the transmitter and receiver ICs are driven by Buffered Line Logic (BLL) Output Stages. BLL drivers [49], shown in Fig. 3.9, provide differential outputs capable of delivering ECL swings into AC-coupled 50 Ω loads through 0.1 μF capacitors. The BLL output impedance is matched to 50 Ω. The external 150 Ω shunt resistors in Fig. 3.9 sets the internal DC bias of the BLL output circuit at 4.1 V.

3.2.2.1. Transmitter IC Operating Principles *

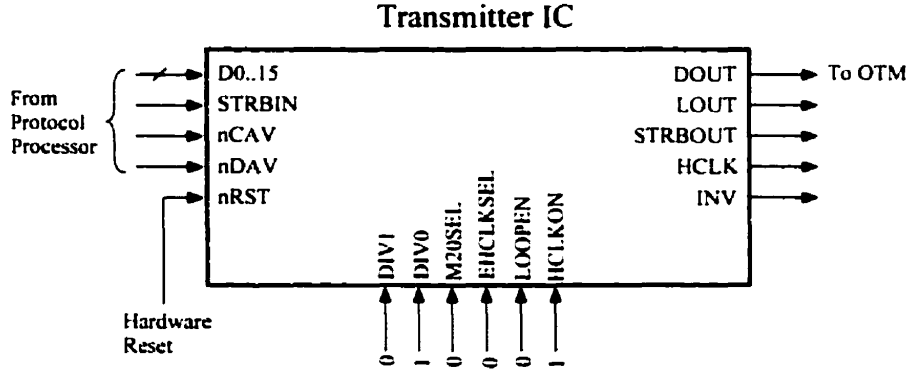


Fig. 3.10. Transmitter IC External Connections and Settings.

Fig. 3.10 shows the external connections and settings of the transmitter IC. The transmitter IC is configured to input 16-bit parallel frames at 40 MHz, by setting the control inputs (from the Protocol Processor) DIV0 to logic 1, and DIV1 and M20SEL to logic 0. The 40 MHz frame clock is supplied on the differential H50 input STRBIN. The transmitter IC can accept parallel frame clock rates from 7.5 to 75 MHz. Inputs DIV1 and DIV0 select the operating frequency range of STRBIN, as specified by the 2nd column in Table 3.1.

DIV1 / DIV0	Parallel Frame Rate (MHz)	Logical Serial Data Rate (Mbit/s)	Encoded Serial Data Rate (Mbit/s)	Freq. Divide
0 0	42.0 - 75.0	672 - 1200	840 - 1500	1
0 1	21.0 - 50.0	336 - 808	420 - 1010	2
1 0	11.0 - 25.0	168 - 404	210 - 505	4
1 1	7.5 - 13.0	120 - 202	150 - 253	8

Table 3.1. Typical Operating Ranges for 16 Bit Mode [49].

In Fig. 3.10, the parallel data from the Protocol Processor is received through the

PECL data inputs D0..15. Inputs nCAV and nDAV specify the frame type to be transmitted. The transmitter IC is configured to output the serial data stream on the differential BLL output DOUT, by setting the serial data output select LOOPEN=0. The serial clock appears on the differential BLL output HCLK, since the serial clock output enable HCLKON=1. The reset input nRST is connected to a DIP switch. It is used for manual initialization of the NIC, when held low for at least 5 frame clock cycles.

The transmitter and receiver ICs employ *Conditional Invert with Master Transition* (CIMT) line encoding [48] for serial data transmission. This scheme maintains the DC balance of the data stream, and facilitates clock recovery and frame alignment (i.e., establishing the boundaries of the serial frames) at the receiver IC. The encoding algorithm creates a 20-bit frame by appending 4 control bits to the 16 data bits. The resulting 20-bit frame may be inverted as necessary to maintain the DC balance of the serial bit stream. In Table 3.1, the 3rd column represent the logical data rate without CIMT encoding and the 4th column denote the physical serial transmission rate due to CIMT encoding. The CIMT coding scheme has been adopted by the Serial-HIPPI (High Performance Parallel Interface) and by the IEEE's SCI-FI (Scalable Coherent Interface - Fiber) standards [48].

3.2.2.2. Clock Generation and Phase-Locked Loop *

Referring to the block diagram in Fig. 3.6, the Phase Locked Loop (PLL) module within the transmitter IC performs clock multiplication and generates all the internal frame and serial clock signals. In 16-bit mode, the PLL locks onto the 40 MHz parallel frame clock STRBIN, which is multiplied up by 20 (16 data bits and 4 control bits) to produce the 800 MHz serial clock.

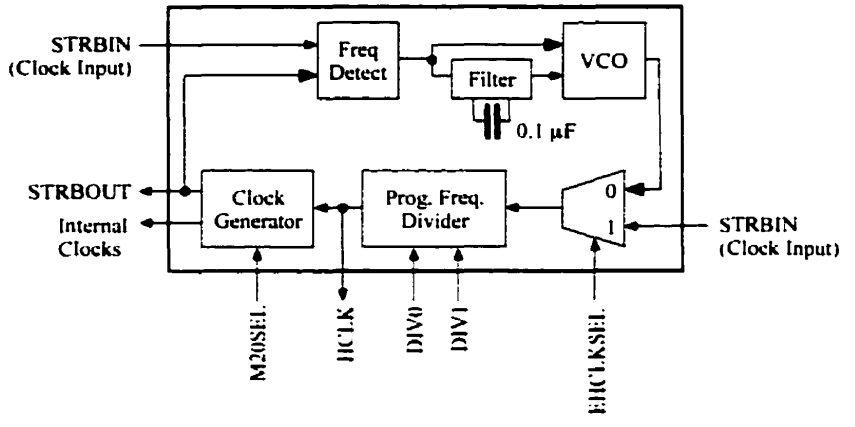


Fig. 3.11. Transmitter IC Phase Locked Loop Module.

The transmitter IC PLL module is shown in Fig. 3.11. It consists of a Frequency Detector, Loop Filter, Voltage-Controlled Oscillator (VCO), Programmable Frequency Divider and Clock Generator. The *Frequency Detector* compares STRBIN with the internal frame clock generated by the Clock Generator. The outputs pass through a *Loop Filter* that controls the center frequency of the VCO output. The loop filter requires an external 0.1 μF capacitor, placed across pins CAP0 and CAP1, as shown in Fig. 3.6. The *VCO* clock is divided by $N \in (1,2,4,8)$, which is determined by the operating frequency band and the 5th column in Table 3.1. This divided frequency is used as the serial clock and is accessible through the differential BLL output HCLK, when the serial clock output enable HCLKON=1. The *Clock Generator* completes the feedback loop and creates all the internal clock signals. The retimed frame clock output STRBOUT is phase-locked to STRBIN and has a frequency of HCLK/20. An external high-speed clock can be used to bypass the VCO clock by setting the external serial clock input enable EHCLKSEL=1. In this case, the external serial clock is provided on input STRBIN and is used directly to output the serial data.

3.2.2.3. Frame Formats *

According to the CIMT encoding algorithm, the data inputs D0..15 are formatted into a 20-bit frame containing a 16-bit Data Field, *DF*, and a 4-bit Control Field, *CF*. Let DF_i denote bit $i \in (0..15)$ in the Data Field and CF_j denote bit $j \in (0..3)$ in the Control Field. In the 20-bit frame, DF_0 is the least significant bit and CF_3 is the most significant bit. The 4-bit Control Field, generated by the *Control Field Encoder* in Fig. 3.6, encodes the status of the current frame (frame type, inversion, transmitter IC settings) and enables the receiver IC to correctly decode the data stream at the destination NIC. Bits $CF_{1..2}$ of the Control Field form a *Master Transition* (rising edge "01" or falling edge "10") which serves as a fixed timing reference for the receiver IC clock recovery circuit, as described in Section 3.2.3.2. The *Data Field Encoder* in Fig. 3.6 generates the 16-bit Data Field according to the frame format selected for serial transmission. Fig. 3.12 illustrate the 3 supported frame types: Data Frame, Control Frame, and Fill Frame. The Protocol Processor setting input signals nCAV and nDAV specify the frame type to be transmitted.

Data Frames, specified by nDAV=0, are used to encode the 16-bit parallel data D0..15 for transmission. The Data Field Encoder places the parallel data inputs D0..15 into the 16-bit Data Field, which is appended with the Control Field $CF_{0..3} = 1101$ to form the 20-bit Data Frame in Fig. 3.12.

Control Frames, specified by nCAV=0, are intended for the transmission of

application specific control information defined by the user, such as packet headers, link or system control packets. The 14-bit control information is supplied on parallel data inputs D0..13. The Data Field Encoder places inputs D0..6 into $DF_{0..6}$ and inputs D7..13 into $DF_{9..15}$ of the Data Field, as shown in Fig. 3.12. Bits $DF_{7..8}$ of the Data Field are set to "01" to distinguish between Control and Fill Frames. The Data Field is appended with $CF_{0..3} = 0011$ to form the 20-bit Control Frame in Fig. 3.12.

Fill Frames are sent during link startup to establish frame synchronization and when no user data is being transmitted to allow the receiver IC to maintain frequency and phase lock. If both nCAV and nDAV are at logic 1, the Control Logic assumes the link is unused and triggers the Data Field Encoder to set $DF_{0..7} = 11111111$ and $DF_{8..15} = 00000000$. The Control Field Encoder generates $CF_{0..3} = 0011$. The 20-bit Fill Frame in Fig. 3.12 has a 50% duty cycle and a single rising edge at the Master Transition $CF_{1..2}$.

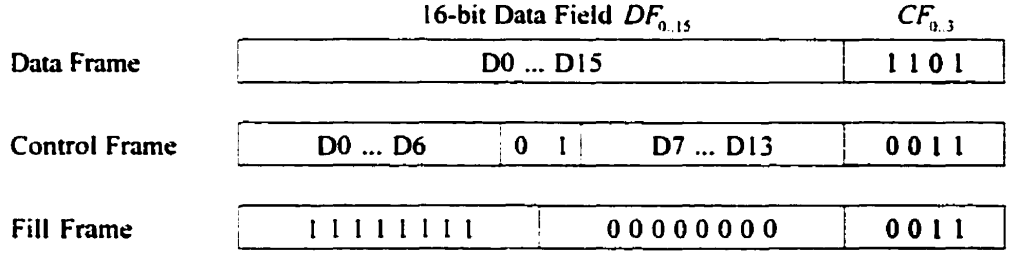


Fig. 3.12. Supported Frame Formats.

3.2.2.4. Data Serialization *

The parallel data inputs D0..15 and the control inputs nDAV, nCAV and nRST are latched on the rising edge of the frame clock STRBIN. Based on input signals nDAV and nCAV, the *Control Logic* in the block diagram of Fig. 3.6 determines the outputs of the Data Field Encoder and the Control Field Encoder. The *Data Field Encoder* supplies the Data Field $DF_{0..15}$ and the *Control Field Encoder* generates the Control Field $CF_{0..3}$ in one of the formats shown in Fig. 3.12. The *Frame Mux* accepts the parallel outputs from the Data Field and Control Field Encoders, and multiplexes the 20-bit parallel frame to 1 bit at the serial clock rate of 40 MHz × 20 bits = 800 Mbit/s. The Frame Mux serially outputs the 16-bit Data Field first, from DF_0 through DF_{15}, followed by the 4-bit Control Field, from CF_0 through CF_3, to the Line Encoder. At the same time, the *Sign* circuitry computes the cumulative sign of the parallel frame (i.e., number of 1's versus number of 0's). The sign is "positive" if there are more 1's than 0's in the 20-bit frame, and is "negative" if otherwise.

This information is used by the Line Encoder to maintain DC balance for the transmission line.

The *Line Encoder* in Fig. 3.6 compares the signs of the current and previous frames, and determines whether frame inversion should occur. The current serial frame is inverted by the Line Encoder if the signs are the same, but remains unchanged if the signs differ. No inversion is performed for Fill Frames. The conditional inversion prevents static input data patterns from generating false lock points in the transmitted serial data stream, and brings the line closer to the desired 50% duty cycle. The conditional inversion of the CIMT line encoding is illustrated in Tables 3.2 and 3.3. When the current serial frame is inverted, output INV=1.

Data Frame	Data Field	Control Field
Non-inverted	D0...D15	1 1 0 1
Inverted	$\overline{\text{D0...D15}}$	0 0 1 0

Table 3.2. CIMT Line Encoding for Data Frame.

Control Frame	Data Field			Control Field
Non-inverted	D0...D6	0 1	D7...D13	0 0 1 1
Inverted	$\overline{\text{D0...D6}}$	1 0	$\overline{\text{D7...D13}}$	1 1 0 0

Table 3.3. CIMT Line Encoding for Control Frame.

The CIMT encoded serial frame is sent to the *Output Select* unit in Fig. 3.6, which is configured to output the serial frame on the differential BLL port DOUT (LOOPEN=0).

The timing characteristics of the serialization of a parallel 16-bit data frame are described in Appendix A.

3.2.3. Receiver Module

The Receiver Module within the NIC in Fig. 3.4 performs the inverse function of the Transmitter Module. It demultiplexes the 8-bit datapath at 800 MHz from the Optical Transceiver Module (OTM) to an 128-bit datapath at 40 MHz. Referring to Fig. 3.13, a complete Receiver Module consists of 8 Hewlett-Packard HDMP-1014 Gigabit receiver ICs [48][49] and 34 Motorola 10H350 PECL-CMOS translators [50].

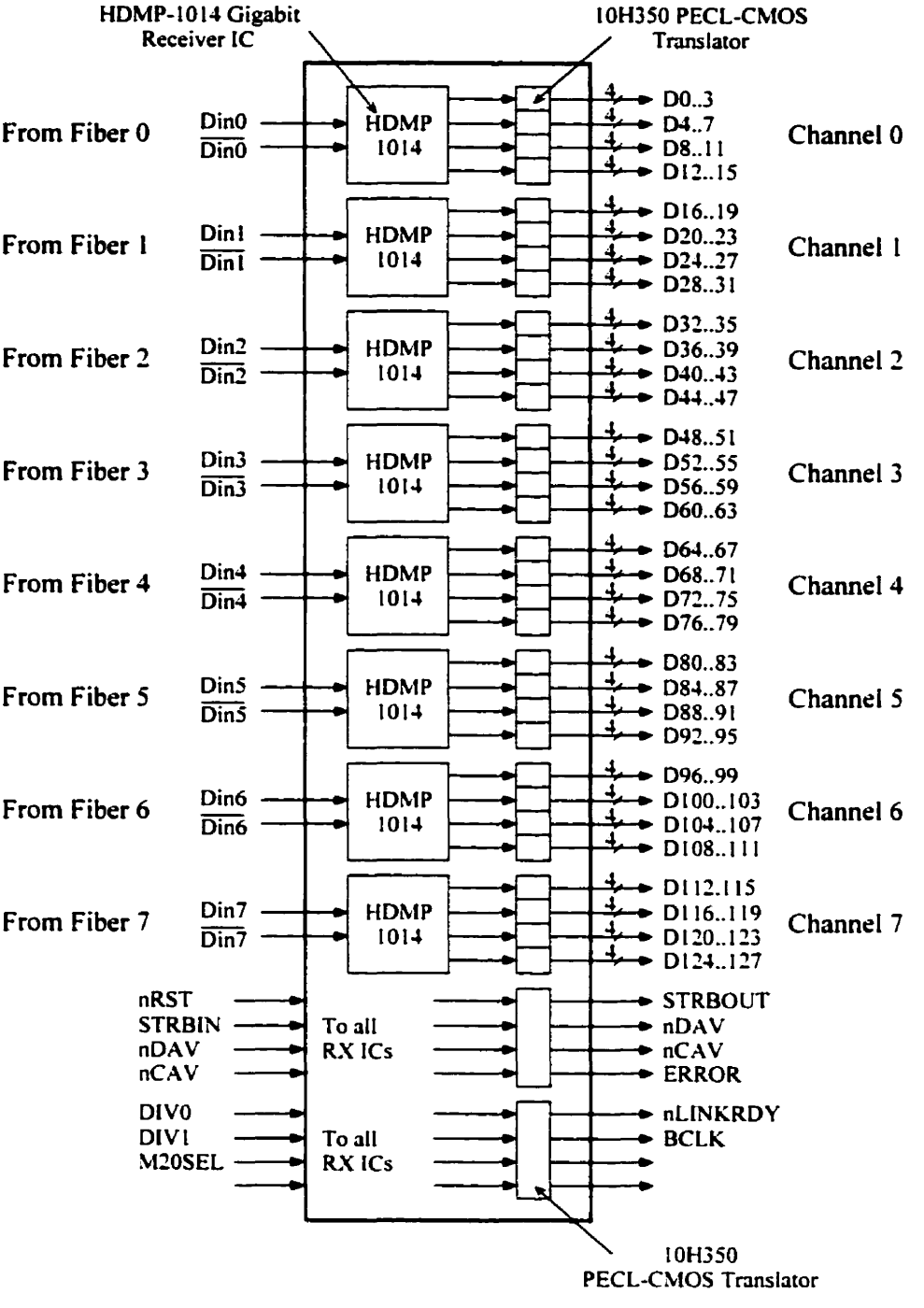


Fig. 3.13. Complete Receiver Module.

Each receiver IC has a 80-pin 0.80 mm lead pitch rectangular QFP package and can deserialize the CIMT encoded serial stream at 800 MHz to generate 16 bits at 40 MHz. Each PECL-CMOS translator IC has a 16-pin DIP package and operates on 4 differential PECL inputs to generate 4 single-ended CMOS outputs.

The deserialization of the 8-bit datapath to 128 bits require 8 receiver ICs, as shown in Fig. 3.13. Each receiver IC at the destination NIC deserializes 1 of the 8 bit-serial data streams from the Optical Transceiver Module (OTM) to form 16 bits of the 128-bit parallel datapath. Each receiver IC requires 4 PECL-CMOS translators to convert its 16-bit data output, for a total of 32 translators. Additional translators are used to convert the receiver IC control signal outputs. Hence, the complete Receiver Module requires 8 receiver ICs and 34 PECL-CMOS translators, as shown in Fig. 3.13.

The receiver IC [48][49] within the Receiver Module contains circuitry for clock recovery and generation, serial-to-parallel conversion, and data and link status extraction, as shown in Fig. 3.14. The receiver IC electrical I/O are summarized in Appendix A. Readers who wish a broad overview may proceed to Section 3.2.4.

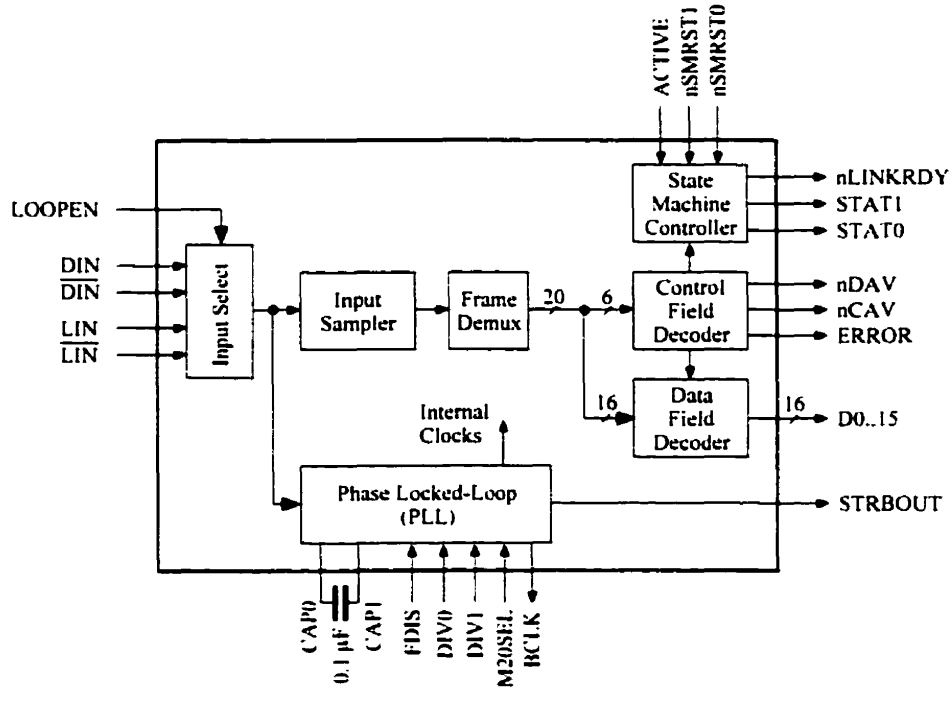


Fig. 3.14. HDMP-1014 Gigabit Receiver IC Block Diagram.

3.2.3.1. Receiver IC Operating Principles *

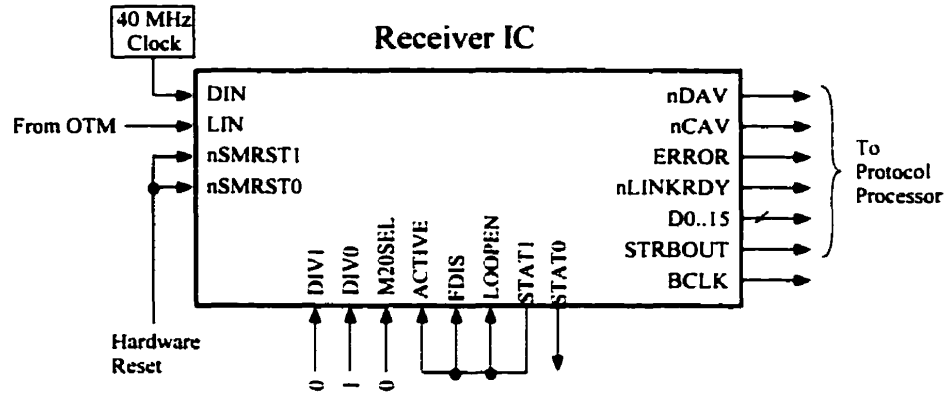


Fig. 3.15. Receiver IC External Connections and Settings.

Fig. 3.15 shows the external connections and settings of the receiver IC. The serial data from the Optical Transceiver Module (OTM) is received through the differential H50 input LIN. A reference clock at the frame frequency of 40 MHz is connected to the differential H50 input DIN. The receiver IC is configured to output 16-bit parallel frames at 40 MHz on the data outputs D0..15, by setting the control inputs (from the Protocol Processor) DIV0 to logic 1, and DIV1 and M20SEL to logic 0. The serial clock and frame alignment are extracted from the incoming data stream by the Phase Locked Loop (PLL) module within the receiver IC. In Fig. 3.14, the PLL is controlled by an integrated State Machine Controller (SMC), which is described in Section 3.2.3.3. The SMC reset inputs nSMRST0 and nSMRST1 are connected to a DIP switch for manual initialization of the NIC. The SMC output STAT1 drives the inputs ACTIVE, FDIS and LOOPEN.

3.2.3.2. Clock Recovery and Phase-Locked Loop *

Recovery of the serial clock from the incoming serial signal and the generation of internal clock signals are performed by the PLL, which is based on a binary-quantized phase detector [48][51]. In Fig. 3.16, the receiver PLL consists of the Frequency-Phase Detectors, Loop Filter, Voltage Controlled Oscillator (VCO), Programmable Frequency Divider and Clock Generator.

From Fig. 3.14, the *Input Select* unit selects between 2 sets of high-speed inputs to the receiver IC. The serial input DIN is used when input select LOOPEN=0 and LIN is used when LOOPEN=1. The output of the Input Select unit drives the *Frequency-Phase*

Detectors in Fig. 3.16, which compare the frequency and phase of the serial input to the internal serial clock generated by the Clock Generator. In Fig. 3.16, the FDIS control signal selects between the frequency or phase detector outputs. The frequency detector determines the frequency of the incoming serial data by locking onto the Master Transition (i.e., bits $CF_{1..2}$ of the Control Field). Once frequency lock is accomplished, the phase detector is used to monitor frame synchronization and to receive data. As shown in Fig. 3.14, a 0.1 μF capacitor must be placed across pins CAP0 and CAP1. This loop filter converts the error output of the Frequency-Phase Detectors to a control signal for the *VCO*. This control signal determines the direction (increase or decrease) and the magnitude of the frequency change of the VCO.

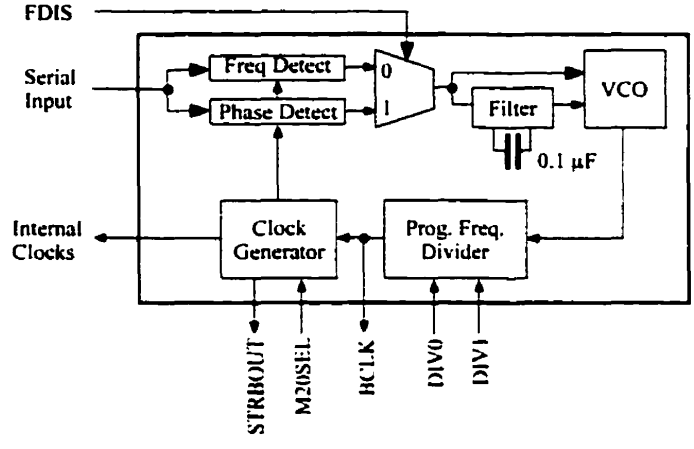


Fig. 3.16. Receiver IC Phase Locked Loop Module.

The *Programmable Frequency Divider* in Fig. 3.16 divides the VCO clock by $N \in (1,2,4,8)$, which is set according to the frequency band selected using inputs DIV1 and DIV0, as shown in Table 3.1. This divided frequency, BCLK, is the recovered serial clock. The *Clock Generator* completes the feedback loop, and generates the internal clock signals for data sampling and demultiplexing. In 16-bit mode, the Clock Generator divides the serial clock BCLK by 20 (16 data bits and 4 control bits) to form the recovered frame clock output STRBOUT.

3.2.3.3. Link Startup and Synchronization *

In Fig. 3.14, the receiver IC contains a State Machine Controller (SMC), which performs the functions of link startup, frame synchronization and error checking. The CIMT line code provides a guaranteed transition at a fixed location in every frame. During link startup, the PLL determines the frequency of the incoming serial data and locates the

Master Transition, which is then used to establish and to monitor frame synchronization.

The SMC monitors the lock conditions, and controls the Input Select unit and the PLL through its outputs STAT0 and STAT1, which denote the current state of the SMC. As shown in Fig. 3.15, output STAT1 is externally connected to control inputs LOOPEN, FDIS and ACTIVE. By providing the SMC status outputs, the receiver IC can be configured externally to operate in one of the four supported link configurations [49].

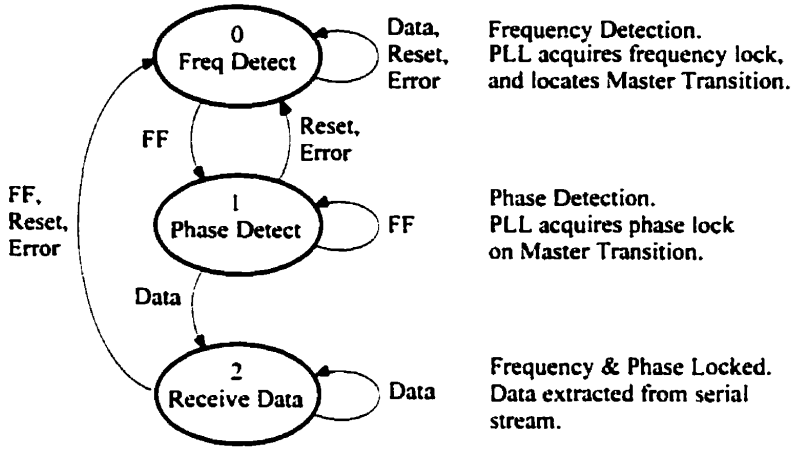


Fig. 3.17. State Machine Controller State Diagram.

The state diagram of the SMC is shown in Fig. 3.17. The SMC goes into the Frequency Detect state when the receiver IC is reset (i.e., reset input nSMRST0=0 or nSMRST1=0) or when errors are detected (output ERROR=1). The receiver IC uses DIN as the serial input, and the PLL attempts frequency acquisition on the 40 MHz reference clock on input DIN. The periodic reference clock signal emulates the Fill Frame of Fig. 3.12. The rising clock edge serves as the Master Transition to establish an unambiguous frame reference. Upon frequency lock, the SMC goes into Phase Detect state. The receiver IC switches the serial input from the reference clock on input DIN to the data stream on input LIN. The Master Transition is monitored to maintain an accepted phase error of ±22.5°. When frequency and phase locks are achieved, the SMC proceeds into the Receive Data state, where the receiver IC decodes the serial input and outputs valid data.

A detailed description of the SMC is provided in Appendix A.

3.2.3.4. Data Deserialization *

Referring to the receiver IC block diagram in Fig. 3.14, the serial data input on LIN is converted into a serial bit stream by the *Input Sampler*, using the recovered serial clock from the PLL. The sampled 20-bit serial frame is sent from the Input Sampler to the Frame

Demux. Recall that DF_i denote bit $i \in (0..15)$ in the Data Field and CF_j denote bit $j \in (0..3)$ in the Control Field. In the 20-bit frame DF_0 is the least significant bit and CF_3 is the most significant bit. The *Frame Demux* deserializes the sampled serial frame into a 20-bit wide parallel word. $DF_{0..15}$ is sent to the Data Field Decoder. $CF_{0..3}$ and $DF_{7..8}$ are sent to the Control Field Decoder, as shown in Fig. 3.14.

By examining bits $CF_{0..3}$ and $DF_{7..8}$, the *Control Field Decoder* can determine the frame type and its conditional inversion status. When a Data Frame is received, output signal nDAV=0. When a Control Frame is received, output signal nCAV=0. A Fill Frame is indicated by nCAV=1 and nDAV=1. The receiver IC detects the loss of a Master Transition or an invalid Control Field code as a frame error, and asserts output ERROR=1. The Protocol Processor monitors the receiver IC output signals nDAV, nCAV and ERROR, and interprets the parallel output data accordingly.

The Control Field Decoder controls the output of the *Data Field Decoder*. If an inverted Data or Control Frame is detected by the Control Field Decoder, the Data Field Decoder will invert $DF_{0..15}$. For a Data Frame, $DF_{0..15}$ is sent directly to the parallel data outputs D0..15. For a Control Frame, $DF_{0..6}$ appear on outputs D0..6 and $DF_{9..15}$ appear on outputs D7..13., thus forming the 14-bit parallel control word.

The deserialization timing of the receiver IC, with respect to a 20-bit serial frame, is described in Appendix A.

3.2.4. Optical Transceiver Module

The Optical Transceiver Module (OTM) within the NIC in Fig. 3.4 functions as the interface between the electrical and optical domains. The OTM can support an aggregate bandwidth of 6.4 Gbit/s to and from the NIC. Referring to Fig. 3.18, a complete OTM contains one Motorola Optobus IC [17][52] and four Motorola 10E416 PECL signal buffers [53]. The Optobus IC has a 14x14 96-pin PGA package, with 10 differential inputs, 10 differential outputs and 56 power-ground pins. Each PECL signal buffer IC has a 28-pin PLCC (plastic leaded chip carrier) package, with 5 differential inputs and 5 differential outputs. The 10E416 IC is used as a high bandwidth amplifier and its internal gain stages can restore the inputs to full PECL swing (3.3-4.1 V).

On the transmit side, the OTM performs electrical-to-optical (E/O) conversion on the serial bytes from the local Transmitter Module. The differential PECL-level serial bytes pass through the 10E416 signal buffer ICs before entering the Optobus transceiver and are then transmitted over the outgoing parallel fiber ribbon to the central Switch Core PCB. On the receive side, the serial bytes are received over the incoming parallel fiber ribbon and the

OTM performs optical-to-electrical (O/E) conversion on the serial bytes from the Switch Core PCB. The differential Optobus output signals are restored to PECL levels by the 10E416 signal buffer ICs and sent to the local Receiver Module. Readers who wish a broad overview may proceed to Chapter 4.

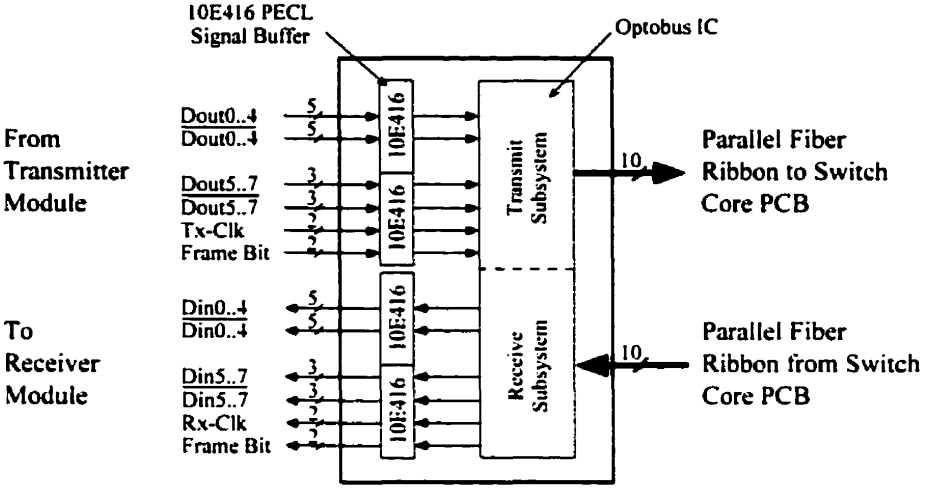
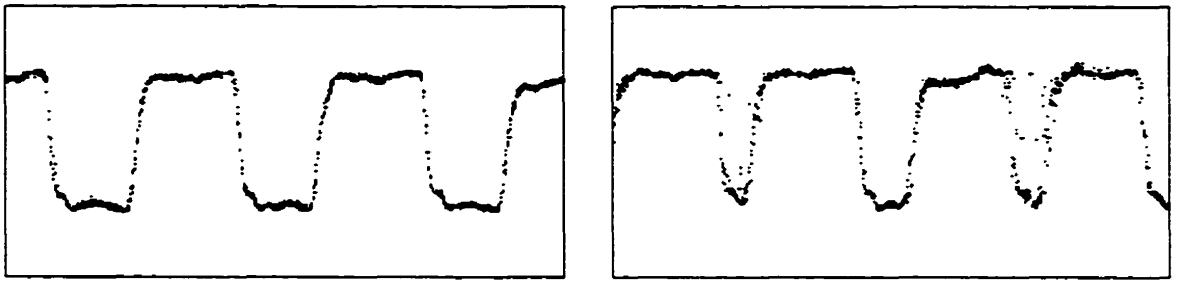


Fig. 3.18. Complete Optical Transceiver Module on the NIC.

3.2.4.1. Motorola Optobus IC *

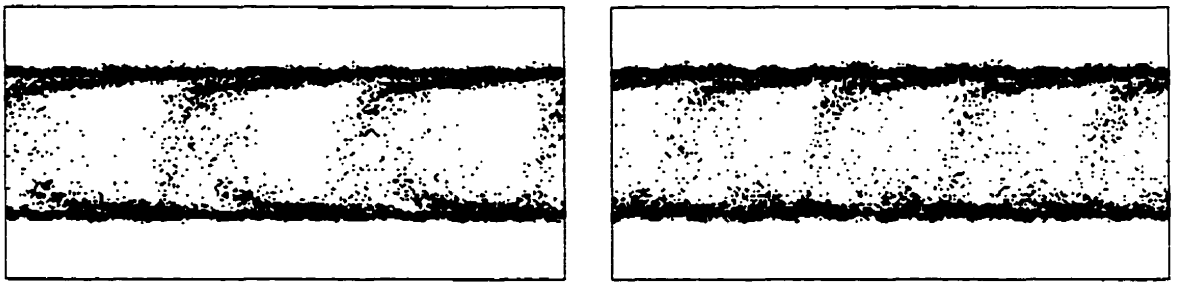
The Motorola Optobus [17][52] is a bi-directional point-to-point optical datalink based on 850 nm wavelength Vertical Cavity Surface Emitting Laser (VCSEL) technology. The Optobus supports 10 transmit and 10 receive channels, on two connectorized 10-bit multimode parallel fiber ribbons. The channels are asynchronous and DC-coupled, and each one has a bandwidth of 800 Mbit/s and a bit error rate of 10^{-14} [17][52]. The link jitter is specified at 650 ps and inter-fiber skew is specified at 200 ps for distances up to 300 m [54]. The Optobus consumes 1.6 W of power when operated from a $V_{CC} = 5$ V supply [17]. Figs. 3.19 and 3.20 show the waveforms and eye diagrams of the Optobus datalink measured experimentally at various data rates [29].

The parallel fiber ribbon contains 10 graded-index multimode fibers. Each fiber has a 62.5 μm diameter core in a 125 μm diameter cladding. The pitch between fibers is 250 μm. Optical attenuation is rated at 4 dB/km. The Optobus uses a molded-plastic light-guide technology [55] to couple light to and from the fibers, through MT connectors [56]. Fig. 3.21 shows the parallel fiber ribbon output as observed using a CCD (charge-coupled device) camera.



(a) (b)

Fig. 3.19. Measured Waveforms of Optobus Datalink at
(a) 600 Mbit/s, and (b) 800 Mbit/s [29].



(a) (b)

Fig. 3.20. Measured Eye Diagrams of Optobus Datalink at
(a) 600 Mbit/s, and (b) 800 Mbit/s [29].

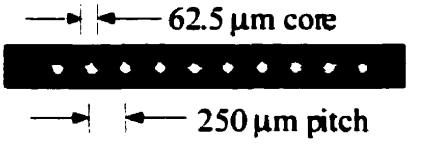


Fig. 3.21. Optobus Parallel Fiber Ribbon Output.

The Optobus is divided into independent transmit and receive subsystems, as shown in the functional diagram of Fig. 3.22. The transmit subsystem consists of the CMOS laser driver circuit and laser-to-fiber interface, which houses the 1x10 VCSEL array. The receive subsystem consists of the fiber-to-detector interface, which houses the 1x10 photodiode array, and the bipolar receive circuit. The 4 distinct components in Fig. 3.22 are surface mounted onto the multi-chip module (MCM) substrate.

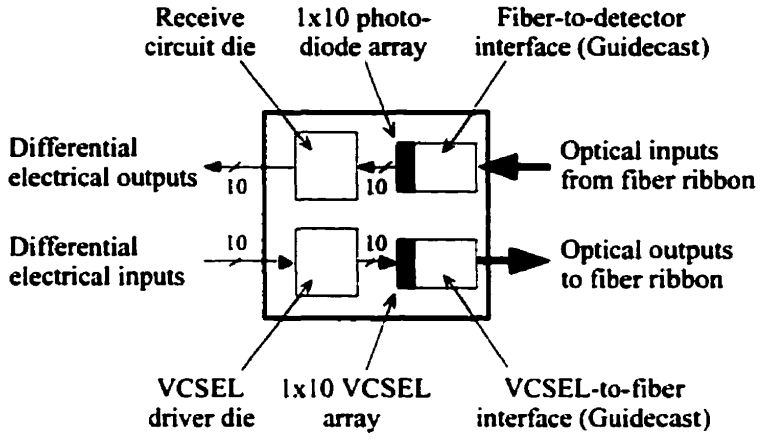


Fig. 3.22. Functional Diagram of the Optobus Module.

3.2.4.2. Optobus IC Operating Principles *

Optical signals are coupled to and from the fiber ribbons using the "bending of electrons" approach. The 1x10 VCSEL array and 1x10 photodiode array are fabricated as discrete optoelectronic dies. The lasers and photodiodes in the arrays are spaced 250 μm apart, which matches exactly the fiber pitch in the parallel fiber ribbon. The schematic of a single optical interface, called the *Guidecast* submount [17][52], is shown in Fig. 3.23. The Guidecast submount positions the optoelectronic die (i.e., the laser array or detector array) perpendicular to the MCM substrate, such that the fibers are aimed directly at the array without intervening optical elements.

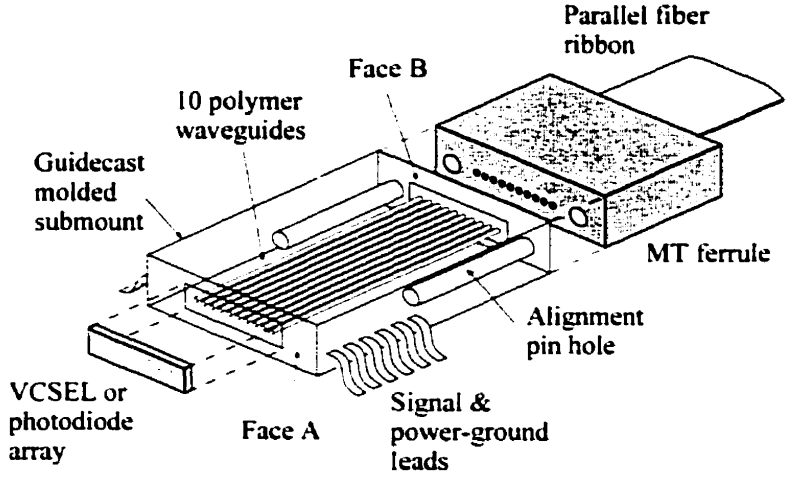


Fig. 3.23. The Guidecast Submount Optical Interface (Adopted from [17]).

The Guidecast submount is a rectangular package molded around 10 polymer waveguides, which are also spaced on a 250 μm pitch. In Fig. 3.23, the waveguides extend the length of the submount from Face A to Face B. The VCSEL or photodiode array is flip-chip bonded to signal and power-ground pads on Face A (pads not shown in Fig. 3.23). These signal and power-ground pads are routed within the submount to external copper leads protruding from the side of the submount (routing not shown in Fig. 3.23). The Guidecast submount is attached onto the Optobus MCM substrate via the external leads, which electrically connect the optoelectronic die to the transmitter laser driver circuit or the receiver detection circuit. Passive alignment of the waveguides in the Guidecast to the parallel fiber ribbon is provided by a pair of alignment pin holes which mate with a MT ferrule at Face B of the submount, as shown in Fig. 3.23.

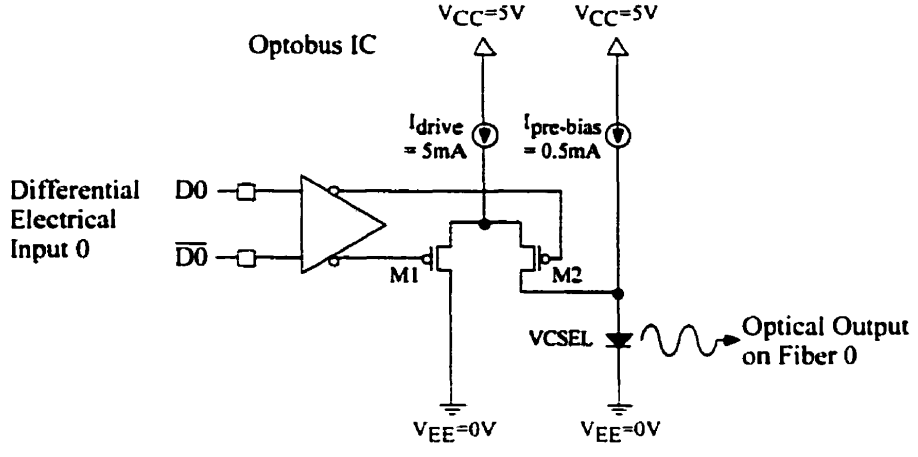


Fig. 3.24. A Single Channel of the Optobus Transmit Subsystem [57].

The transmit subsystem in Fig 3.22 consists of 10 differential electrical inputs and 10 optical output channels. The differential inputs require a minimum swing of 0.25 V with a common mode range from $V_{cc} - 2.25$ to V_{cc} (i.e., 2.75-5.00 V when $V_{cc} = 5$ V), and thus can interface directly with differential PECL inputs (0.8 V swing around 3.7 V). The 850 nm wavelength AlGaAs/GaAs VCSELs are driven by a 10-channel CMOS laser driver circuit [57]. Each VCSEL produces divergent circular beams with a half angle of approx. 8° [17]. A single channel of the transmit subsystem is shown in Fig. 3.24. The laser driver controls the drive current which modulates the VCSEL. The laser is pre-biased at 0.5 mA to lower the laser turn-on delay. Each VCSEL has a peak drive current of 5 mA, lasing threshold of approx. 1.5 mA, and differential quantum efficiency of approx. 0.3 W/A [52]. The peak optical power is 0.8 mW per channel.

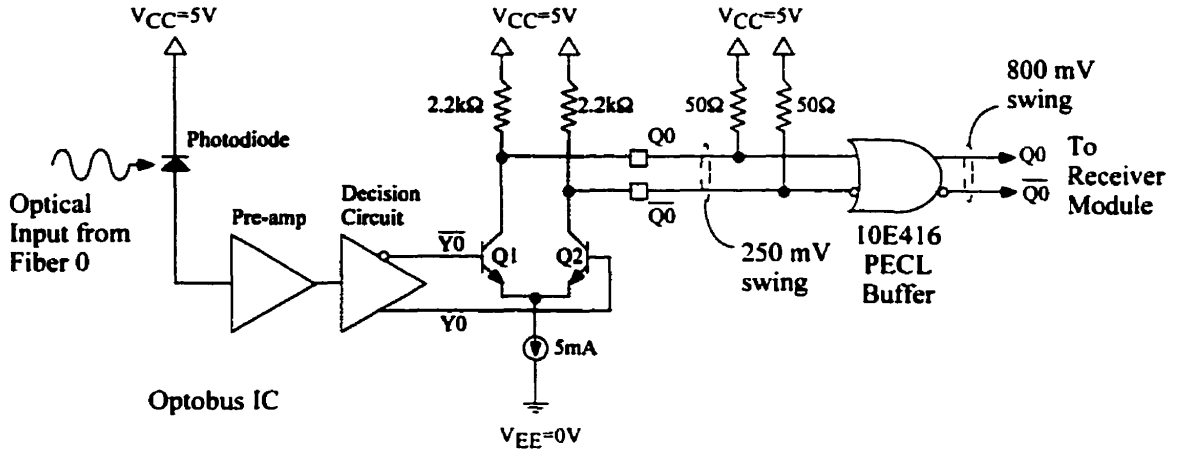


Fig. 3.25. A Single Channel of the Optobus Receive Subsystem [57].

The receive subsystem in Fig. 3.22 consists of 10 optical input channels and 10 differential Current Mode Logic (CML) electrical outputs. A single channel of the receiver circuit contains a GaAs PIN photodiode, a transimpedance pre-amplifier, a threshold decision circuit and the CML output stage, as shown in Fig. 3.25 [57]. The photodiode array is housed in the Guidecast submount of Fig. 3.23, and has an average responsivity of 0.43 A/W [52]. A reference voltage is internally generated to produce a fixed decision threshold of 15 μA of photocurrent, which provides a minimum optical sensitivity of 35 μW.

The differential CML electrical outputs of the Optobus are a type of low voltage differential I/O. The high output impedance CML output stage for Channel 0 is shown in Fig. 3.25. The decision circuit outputs Y0 and $\overline{Y0}$ control transistors Q1 and Q2, which form an inverting differential pair amplifier. The Optobus output pins Q0 and $\overline{Q0}$ are connected to external 50 Ω pull-up resistors to 5.0 V, which also function as terminating resistors. When Y0 = 0 and $\overline{Y0}$ = 1, Q1 turns on and Q2 remains off. The output cell sinks 5 mA of current through output pin Q0 of the differential pair while the complementary output pin $\overline{Q0}$ is at 5 V. The opposite occurs when Y0 = 1 and $\overline{Y0}$ = 0. With external 50 Ω pull-up resistors to 5.0 V, as shown in Fig. 3.25, the CML outputs switch between 4.75 V for a logic low and 5.0 V for a logic high. Due to the constant switching current and the low output voltage swing, CML outputs generate less switching noise than conventional ECL I/O's.

The CML signal (250 mV swing) is restored to PECL level (800 mV swing) by the 10E416 signal buffer ICs, which incorporate 2 stages of gain internally, before feeding to the Receiver Module, as shown in Fig. 3.25.

3.2.5. NIC Datapath Propagation Latency

In this section, the latencies of the NIC's transmit and receive datapaths are analyzed.

The *NIC transmit datapath* extends from the electrical outputs of the Protocol Processor to the optical outputs of the OTM. The 128-bit parallel data from the Protocol Processor propagates through the Transmitter Module (containing CMOS-PECL translator, and transmitter IC) and the OTM (containing PECL signal buffer IC and Optobus IC), before being transmitted to the Switch Core PCB. The latencies of these electronic components in the transmit datapath are given as follows:

- *CMOS-PECL translator* - the input to output propagation delay is 1.3 ns.
- *Transmitter IC* - the parallel input to serial output propagation delay is 2.0 ns; the serialization time is 25 ns (20 bits at 1.25 ns per bit); the total delay is 27 ns.
- PECL signal buffer IC - the input to output propagation delay is 0.35 ns.
- *Optobus IC* - the electrical input to optical output delay is 2.8 ns.

Therefore, the total transmission latency through the NIC's transmit datapath is 31.45 ns or approx. 32 ns.

Similarly, the *NIC receive datapath* extends from the optical inputs of the OTM to the electrical inputs to the Protocol Processor. The serial bytes from the switch core propagate through the OTM (containing the Optobus IC and PECL signal buffer IC) and the Receiver Module (containing the receiver IC, PECL signal buffer IC and PECL-CMOS translator). The latencies of the internal electronic components in the receive datapath are given as follows:

- *Optobus IC* - the optical input to electrical output delay is 2.8 ns.
- *PECL signal buffer IC* - the input to output propagation delay is 0.35 ns.
- *Receiver IC* - the time to assemble 20 serial bits (at 1.25 ns per bit) is 25 ns; the deserialization and decoding time requires an additional frame period of 25 ns; the output delay is 2 ns; the total delay is 52 ns.
- *PECL signal buffer IC* - the input to output propagation delay is 0.35 ns.
- *PECL-CMOS translator* - the input to output propagation delay is 3.5 ns.

Therefore, the total transmission latency through the NIC's receive datapath is 59 ns or approx. 60 ns.

3.3. Summary

This chapter reviewed the transistor-level structure and operation of high-speed PECL logic. The transmission line termination techniques used to compensate for the

analog effects of high-speed signal propagation were discussed.

The detailed design of the workstation NIC was proposed, and is consistent with the overview design presented in [5]. The NIC contains the Protocol Processor, the Transmitter Module, the Receiver Module and the Optical Transceiver Module (OTM). The operating principles of these modules and their internal components were described. The proposed NIC architecture can supply up to 6.4 Gbit/s bandwidth to and from the centralized switch core over the optical datalinks. However, 20% of the link bandwidth is used by the Transmitter Module to implement CIMT line encoding. The CIMT encoding scheme maintains the DC balance of the transmitted serial data stream, and facilitates clock recovery and data alignment by the Receiver Module at the destination NIC.

The Protocol Processor handles the communication protocols in FPGA hardware. It supports a slow, wide bit-parallel CMOS datapath to the Transmitter Module and from the Receiver Module, and provides the control signals to these modules. The Transmitter Module multiplexes the slow, wide CMOS datapath to a fast, narrow PECL datapath. The OTM sends the Transmitter Module output data streams to the switch core and receives the Receiver Module input data stream from the switch core, over the optical datalinks. The Receiver Module demultiplexes the fast, narrow PECL datapath from the OTM to a slow, wide bit-parallel CMOS datapath.

The next chapter will describe the complete design of the 16x16 optoelectronic Switch Core, which processes the data from the workstation NICs.

CHAPTER 4

Optoelectronic Switch Core Design

The design of a 16x16 optoelectronic switch core, which can support an aggregate switching bandwidth of 102.4 Gbit/s is proposed. The switch core architecture is outlined in Section 4.1, and is constructed from high-speed electrical crossbar switches, Optical Transceiver Modules (OTMs) and a Switch Controller on a single Printed Circuit Board (PCB). A bank of eight 16x16 static PECL crossbar switch ICs form the electrical switching elements. Each of the 32 switch core I/O ports has an 8-bit datapath at 800 MHz, and interfaces to an Optical Transceiver Module. The packaging of the 16x16 optoelectronic switch core on a high-speed PCB is described.

Section 4.2 presents the switch core routing configuration and control. The Switch Core PCB is connected to the Controller Workstation through an electrical ribbon cable. Routing information is entered and downloaded to the Switch Core PCB under software control. The Routing Software, running on the Controller Workstation, communicates with the FPGA-based Switch Controller on the Switch Core PCB, using the parallel port protocol. The functions of the Routing Software and its graphical user interface are outlined. The FPGA-based Switch Controller, which implements the communication interface and the control logic for configuring the crossbar switch ICs, is described.

4.1. Design of the Optoelectronic Switch Core

The 16x16 optoelectronic switch core is implemented with parallel crossbar switches on a single PCB, and can deliver 102.4 Gbit/s of aggregate switching bandwidth. The architecture of the optoelectronic switch core is illustrated in Fig. 4.1, and is consistent with the functional description in Section 2.2.2. Each switch core I/O port has a 800 MHz 8-bit datapath, interfaces to an Optical Transceiver Module (OTM), and supports a bandwidth of 6.4 Gbit/s per parallel fiber ribbon to a remote NIC. The OTM performs optical-to-electrical (O/E) and electrical-to-optical (E/O) conversions on the serial bytes between the switch core and the NIC.

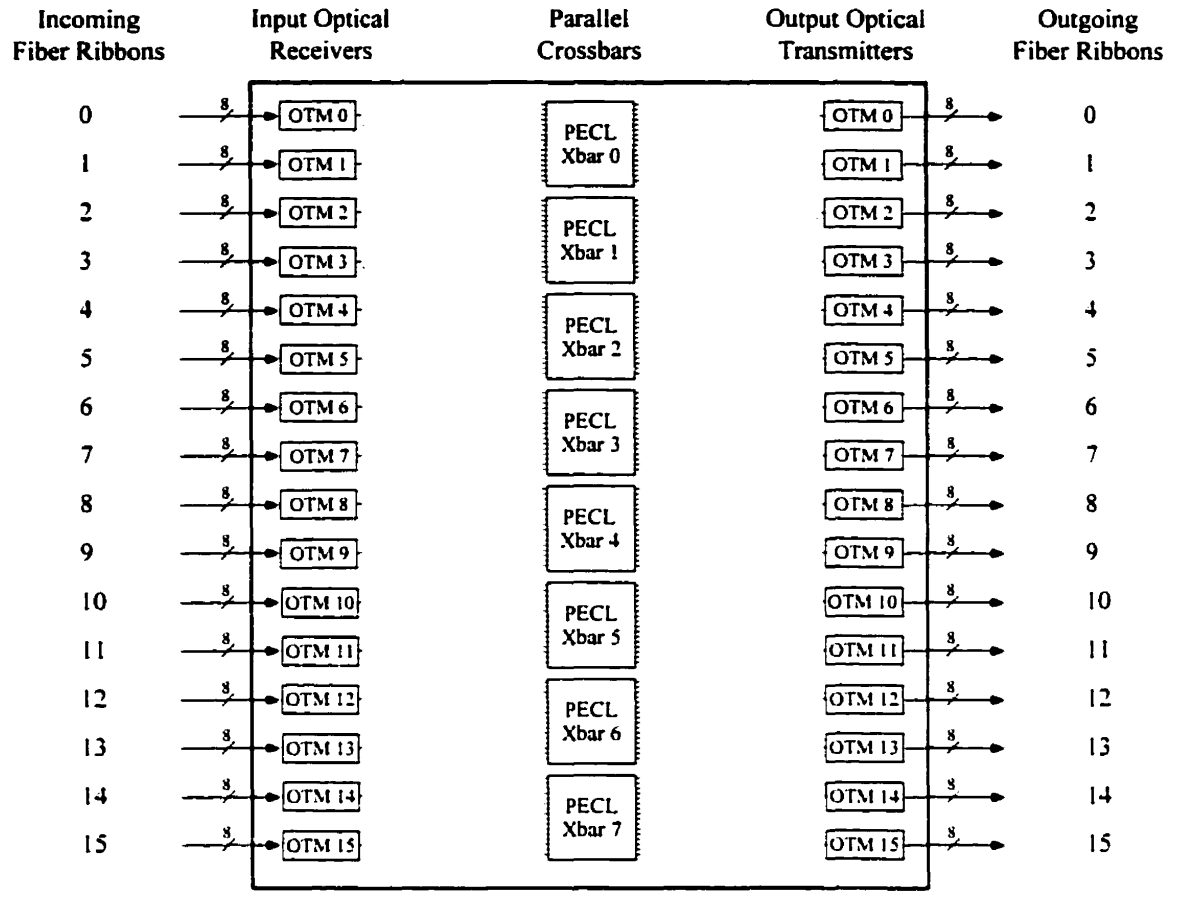


Fig. 4.1. Architecture of the Optoelectronic Switch Core.

The switching function is performed using a bank of eight 16x16 static PECL crossbar switch ICs in parallel, where each crossbar switch IC operates on a 1-bit slice of the byte-wide datapath, as shown in Fig. 4.1. Statically configurable switches are used in the demonstrator for simplicity, but a self-routing switch core can be employed in the long term [5][6][21]. Let $k \in (0..15)$ denote the OTM of a switch core I/O port and $n \in (0..7)$ denote a fiber in the parallel fiber ribbon. At the input to the switch core on the left of Fig. 4.1, OTM k sends the data from fiber n of its parallel fiber ribbon to input port k of crossbar switch IC n. Hence, crossbar switch IC n switches the packets from fiber n of all 16 Optobus datalinks. At the output from the switch core on the left of Fig. 4.1, crossbar switch IC n sends the data from output port k to OTM k for transmission over fiber n of its outgoing parallel fiber ribbon. The data I/O signals of the crossbar switch IC and the OTM in the Switch Core PCB employ differential PECL signaling and the termination techniques in Section 3.1.2 are applied.

On the Switch Core PCB, the electrical switch core is configured by the *Switch*

Controller, which communicates with a dedicated Controller Workstation via an electrical link. Under software control, static routing information is entered and downloaded into the crossbar switch ICs that make up the electrical switch core. Using a CMOS FPGA, the Switch Controller implements the communication interface over the electrical link and the control logic for switch configuration. The control inputs of the crossbar switch IC are CMOS compatible, and can be driven directly by the CMOS FPGA without logic level conversion.

The modules on the Switch Core PCB are constructed using off-the-shelf electronic components, and are described in detail in Sections 4.1.1 and 4.1.2. The descriptions contain detailed technical design information regarding the functionality and operation of the IC components. Readers who wish a broad overview may omit these sections and proceed to Section 4.1.3.

4.1.1. PECL Crossbar Switch IC *

The electrical switch core employs Triquint TQ8017 non-blocking 16x16 crossbar switch ICs, where each port supports a data rate of 1.2 Gbit/s [36]. With 8 ICs, the peak switching capacity is 153.6 Gbit/s. Each crossbar switch IC, has a 132-pin 0.025 in. lead pitch QFP package, with 16 differential PECL inputs, 16 differential PECL outputs, 11 CMOS control inputs and 57 power-ground pins. Power consumption is approx. 1.6 W per IC when operated from a $V_{CC} = 5V$ supply.

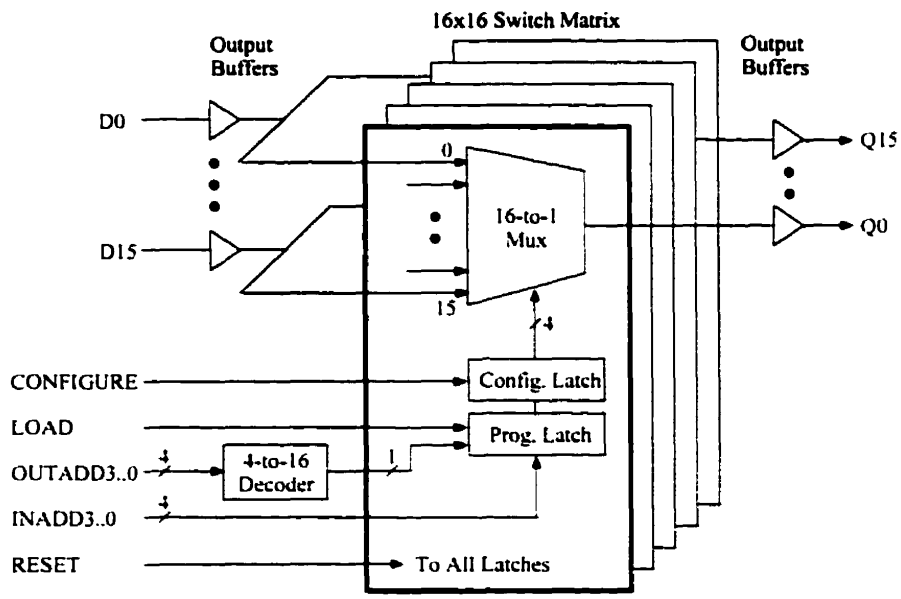


Fig. 4.2. Architecture of the Crossbar Switch IC.

Port Name	Port Type	Description
CNTRL_LVL	GND/Open input	Control inputs signal level select: 0V selects TTL, unconnected selects CMOS
CONFIGURE	CMOS/TTL input	Configuration Latch enable (Active high)
D0..D15, $\overline{D0..D15}$	PECL input	Data input ports 0 to 15
INADD3..0	CMOS/TTL input	Input port address
LOAD	CMOS/TTL input	Program Latch enable (Active high)
Q0..Q15, $\overline{Q0..Q15}$	PECL output	Data output ports 0 to 15
OUTADD3..0	CMOS/TTL input	Output port address
RESET	CMOS/TTL input	Latch reset (Active high)

Table 4.1. Summary of TQ8017 Crossbar Switch IC I/O Ports.

The crossbar switch architecture is shown in Fig. 4.2, and the electrical I/O ports are summarized in Table 4.1. The *Switch Matrix* contains 16 fully independent 16-to-1 multiplexers and allows each output port to accept data from any of the 16 input ports, including broadcast and multicast modes. In Fig. 4.2, one "slice" of the Switch Matrix consists of a 16-to-1 *Multiplexer*, a 4-bit *Program Latch* and a 4-bit *Configuration Latch*. The *Input Buffers* fan-out the data inputs to all 16 multiplexers, such that multiplexer input n is connected to input port n, where $n \in (0..15)$. The *Output Buffers* drive the multiplexer outputs to the differential output pins.

The 16 data input ports D0 to D15 and the 16 data output ports Q0 to Q15 are differential PECL I/O. The remaining inputs are configured for CMOS operation by leaving the CNTRL_LVL pin open. Signal CONFIGURE is the global enable line for the 16 *Configuration Latches*. Signal LOAD is the global enable line for the 16 *Program Latches*. Signal RESET clears all the internal Program and Configuration Latches. The crossbar switch IC operates asynchronously and does not require a clock signal. The switch I/O ports are addressed using the binary form of their port numbers (0 to 15). The 4-bit input port address INADD3..0 is the global data input to all 16 Program Latches. The 4-bit output port address OUTADD3..0 is decoded by the 4-to-16 *Decoder*, which selects 1 of the 16 Program Latches.

The switch configuration is performed by the Routing Software on the Controller Workstation, which drives the address lines and the control inputs using the Switch Controller on the PCB. An *input-output port mapping* is a pair of input port and output port addresses, and is interpreted as input port INADD3..0 being connected to output port OUTADD3..0. The switch is statically configured by loading each output port's Program Latch with the desired input port's 4-bit address. The input port address is then

used as the multiplexer select lines for that output port. The back-to-back latches in each "slice" permit the Program Latch to be modified without interrupting the switching function of the crossbar. The crossbar switch IC timing characteristics and configuration procedures are described in Appendix B.

4.1.2. Optical Transceiver Module *

Each I/O port of the switch core has an 8-bit datapath at 800 MHz, for a total bandwidth of 6.4 Gbit/s per port, and interfaces to an Optical Transceiver Module (OTM). Referring to Fig. 4.3, a complete OTM on the Switch Core PCB contains one Motorola Optobus IC [17] and four Motorola 10E416 PECL signal buffers [53]. These components have been described in detail in Section 3.2.4.

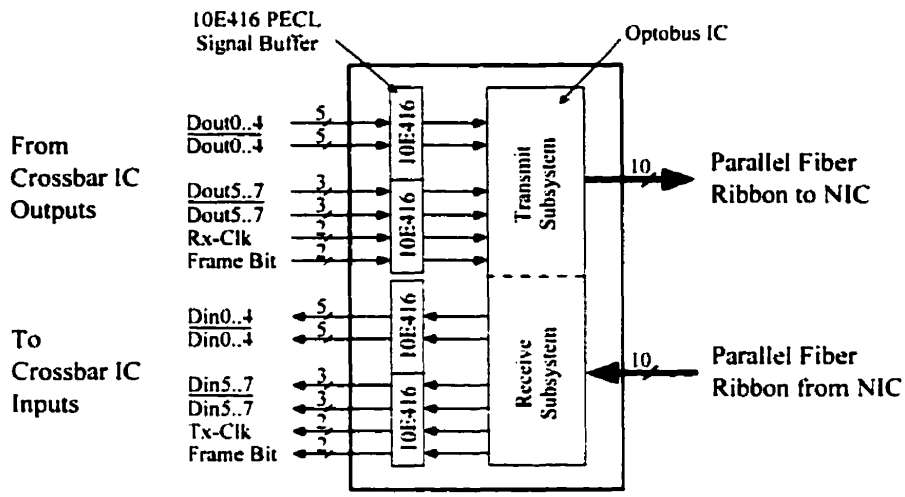


Fig. 4.3. Complete Optical Transceiver Module on the Switch Core PCB.

As mentioned at the beginning of Section 4.1, the Optobus receive subsystem performs O/E conversion on the serial bytes received over the incoming fiber ribbon from the NIC. The differential CML outputs of the Optobus are converted to PECL signals by the PECL signal buffers. The Optobus transmit subsystem performs E/O conversion on the serial bytes sent from the crossbar switch ICs. The differential PECL inputs are restored by the PECL signal buffers before being transmitted by the Optobus.

4.1.3. Propagation Latency

In this section, the latency of the datapath through 16x16 switch core, i.e., from the optical inputs of the switch input port's OTM to the optical outputs of the switch

output port's OTM, is analyzed.

The serial bytes from the sending NIC on the incoming parallel fiber ribbon propagate through the input port's OTM (containing the Optobus IC and PECL signal buffer IC), through the crossbar switch ICs and into the output port's OTM for transmission over the outgoing parallel fiber ribbon to the receiving NIC.

The latencies of these electronic components in the datapath are given as follows:

- Optobus IC - the optical input to electrical output delay is 2.8 ns.
- PECL signal buffer IC - the input to output propagation delay is 0.35 ns.
- Crossbar switch IC - the input to output propagation delay is 2 ns.

Therefore, the total transmission latency through the switch core datapath is approx. 8.3 ns.

4.1.4. Packaging of the Switch Core onto a Single Printed Circuit Board

In this section, the PCB design for the 102.4 Gbit/s switch core connecting 16 PCs is described. An architectural description of this Switch Core PCB is also presented in [5]. The packaging of the optoelectronic switch core onto a single PCB is illustrated in Fig. 4.4. The complete switch core requires 16 Optical Transceiver Modules (OTMs), 8 crossbar switch ICs and one CMOS FPGA-based Switch Controller.

The OTMs are arranged on the sides in 2 columns of 8 modules, and the crossbar switch ICs and the Switch Controller are placed in the center column. Each OTM has a 1-bit differential datapath to and from each of the 8 crossbar switches, connected in perfect-shuffle connections, as shown in Fig. 4.4.

The height of the complete Switch Core PCB can be estimated as follows. An OTM, consisting of one Optobus IC, four 10E416 buffer ICs, terminators and bypass capacitors, measures approx. 2.0 in. high by 3.6 in. wide. Each crossbar switch IC, with terminators and bypass capacitors, measures approx. 2.0 x 2.0 in. The CMOS FPGA also occupies an area of 2.0 x 2.0 in. The spacing between modules is 0.5 in. Hence, the PCB will be approx. 20 in. high when the OTMs are arranged in 2 columns of 8 modules on the sides, and the crossbar switches are arranged in one vertical column, as shown in Fig. 4.4.

The Switch Core PCB must employ the high-speed board design techniques described in [10][58]. A multi-layer printed circuit board can be used with separate power-ground planes dedicated to CMOS and PECL logic. This isolates the CMOS switching noise from coupling into the relatively quiet PECL power-ground planes.

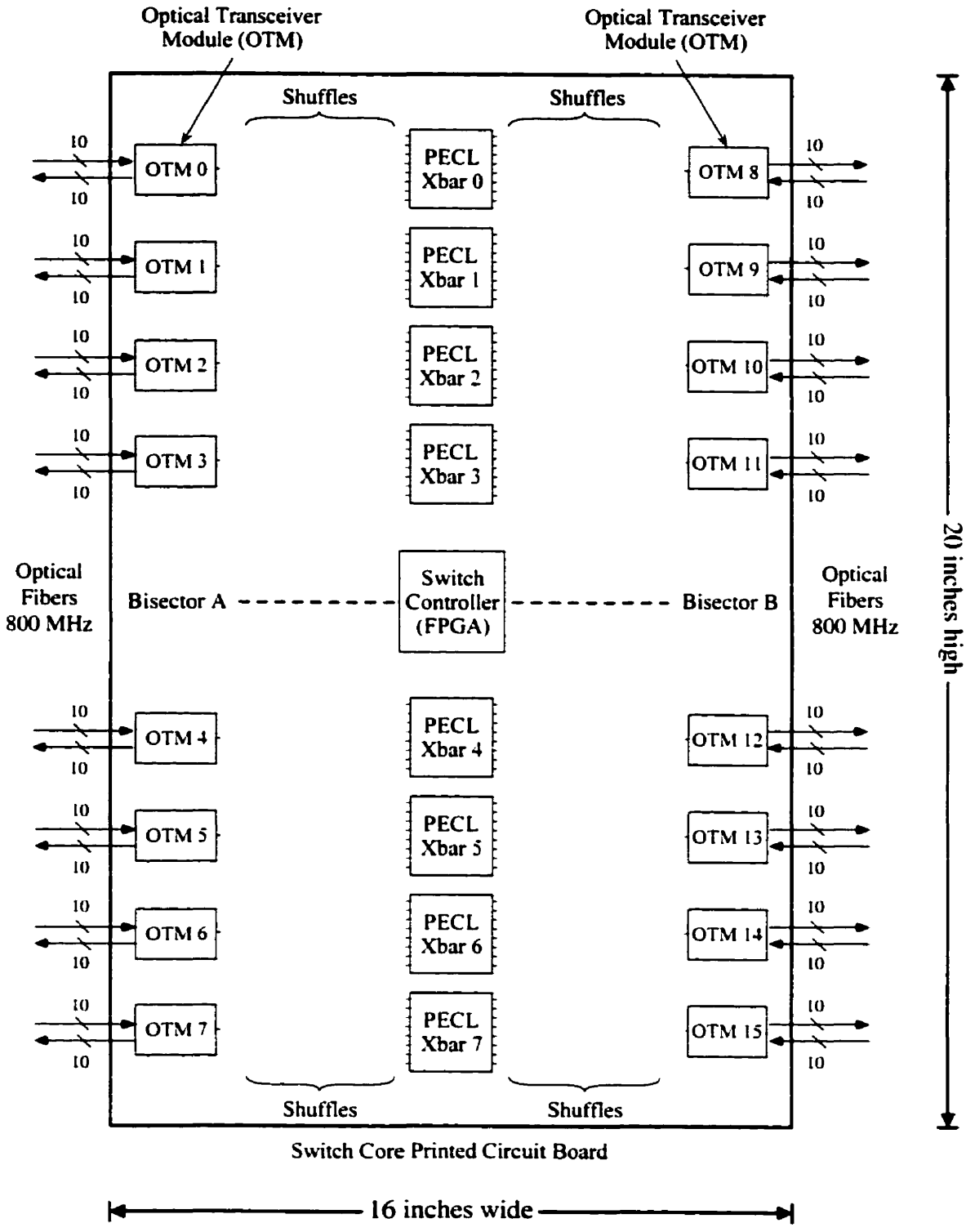
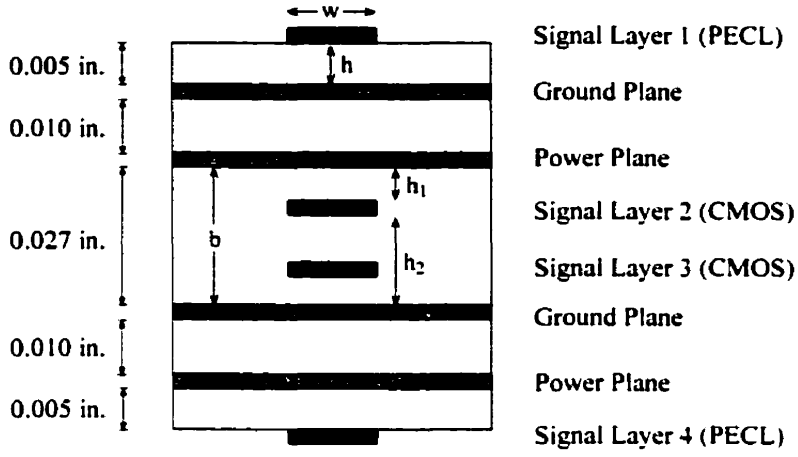


Fig. 4.4. Optoelectronic Switch Core Packaging.

Pairs of solid power-ground planes can be interleaved between pairs of horizontal and vertical signal routing layers. The closely-spaced power-ground planes maximizes their capacitive coupling and reduces power supply noise. The alternating trace orientation in adjacent signal layers reduce coupling and crosstalk between routing layers. To minimize crosstalk between traces on the same routing layer, guard traces, which are grounded at both ends, can be inserted between signal traces, i.e., the datapaths between the crossbar switch ICs and the OTMs of Fig. 4.4.



All layers are 0.0014 inch thick. Total PCB thickness is 0.063 inch.

Fig. 4.5. Stackup for the 8-layer Switch Core PCB.

For maximizing performance, the trace geometry must be controlled to maintain 50 Ω characteristic impedance. Fig. 4.5 illustrates a PCB stackup for the Switch Core PCB. The layer arrangement is symmetric about the center of the PCB. The thickness of all signal and power-ground layers are 0.0014 in. (when 1-oz per sq. ft. of copper is used). The thickness of the PCB is 0.063 in. An outer trace on the surface of the PCB (i.e., signal layers 1 and 4) above a reference power or ground plane is called a *microstrip* conductor and the impedance is approximated by [58]:

$$Z_O = \frac{87}{\sqrt{\varepsilon_r + 1.41}} \ln\left[\frac{5.98h}{0.8w + t}\right] \quad \text{when } 0.1 < \frac{w}{h} < 2.0 \qquad \text{(Eq. 4.1)}$$

where Z_O is the characteristic impedance (Ω)

ε_r is the dielectric constant of the PCB material

w is the width of the trace (inch)

t is the thickness of the trace (inch)

h is the height of the trace above a reference plane (inch)

Referring to Fig. 4.5, the inner traces in signal layers 2 and 3 are sandwiched between 2 reference planes and are called *dual stripline* conductors. The trace impedance is approximated by [58]:

$$Z_O = \frac{30}{\sqrt{\varepsilon_r}} \ln\left[\frac{3.61(2h_1 + t)(2h_2 + t)}{(0.8w + t)^2}\right] \qquad \text{(Eq. 4.2)}$$

$$\text{when } \frac{w}{h_1 + h_2} < 0.35 \text{ and } \frac{t}{h_1 + h_2} < 0.25$$

where h_1 is the distance of the trace from the first reference plane (inch)

h_2 is the distance of the trace from the second reference plane (inch)

Typically, $\varepsilon_r = 4.5$ for epoxy glass FR4 PCB and $t = 0.0014$ in. when 1 oz. per sq. ft. of copper is used. From Eq. 4.1, to obtain a 50 Ω outer trace in layers 1 and 4, we can use trace widths $w = 0.008$ in., placed $h = 0.005$ in. above the reference plane, as shown in Fig. 4.5. From Eq. 4.2, to obtain a 50 Ω inner trace in layers 2 and 3, we can use trace widths $w = 0.008$ in. and distances $h_1 = 0.008$ in. and $h_2 = 0.0175$ in, as shown in Fig. 4.5. These geometries are within the capabilities of current PCB technology. To eliminate reflections and to maintain transmission line impedance matching, all high-speed signal traces, such as the datapaths between the OTMs and the crossbar switch ICs, are end terminated into 50 Ω resistors at the receiver, as described in Section 3.1.2. To minimize skew, all traces in a datapath should have comparable length.

With 0.008 in. wide signal and guard traces, and 0.012 in. trace-to-trace separation, each signal layer supports 25 signal and 25 guard traces per inch of PCB, and this wiring density can be used to estimate the number of signal layers and the width of the Switch Core PCB. The size of the complete Switch Core PCB is computed from Fig. 4.4 as follows. The horizontal bisectors A and B separate the PCB into 2 symmetric halves. Consider OTMs 0 through 7 in the left column of Fig. 4.4. Each OTM sends 8 bits to and receives 8 bits from the crossbar switches. These PECL datapaths between the OTMs and the crossbar switch ICs employ differential signaling and operate at 0.8 Gbit/s. Four bits in each 8-bit wide datapath cross the horizontal bisector A, and the number of signal traces per OTM crossing bisector A is $(4 + 4)$ bits $\times 2$ traces per bit $= 16$ traces. With 8 OTMs on either side of the PCB, the total number of signal traces crossing bisector A (or bisector B) is $16 \times 8 = 128$ traces.

Using 2 signal layers at a density of 25 signal traces per inch per signal layer, the shuffles implementing the traces for the PECL datapaths are (128 traces ÷ 25) ÷2 layers ≈ 3 in. wide per shuffle. The PCB also needs 9.2 in. of width for the 3 columns

of on-board ICs (i.e., sum of two OTM widths and one crossbar switch IC width). Separate power-ground planes are reserved for CMOS and PECL logic, and 2 additional layers are used for routing CMOS signals, as shown in Fig. 4.5. Hence the entire PCB has 8 layers and measures 20 in. high, 16 in. wide and 0.063 in. thick, which is within the limits of current PCB technology [10]. Furthermore, the design has been conservative and has used relatively wide traces. High-performance PCBs can use much finer metal trace widths (0.0005 in.) and up to 40 layers of metal [59], resulting in much smaller PCBs. This analysis is consistent with the architectural projections in [5][21].

4.2. Routing Configuration and Control

The routing configuration of the 16x16 optoelectronic switch core is managed by a combination of software and hardware. As shown in Fig. 4.6, the Switch Core PCB connects to the Controller Workstation's parallel I/O port, through an electrical ribbon cable. On the Controller Workstation, custom *Routing Software* [30] communicates with the Switch Controller on the PCB using the parallel port protocol [60]. The software provides an efficient interface for entering the input-output port mappings of the 16x16 optoelectronic switch core. The Routing Software is described in Section 4.2.1. On the Switch Core PCB, an FPGA-based *Switch Controller* [30] handles the data transfer protocol and implements the control logic for configuring the crossbar switch ICs. The Switch Controller is described in Section 4.2.2.

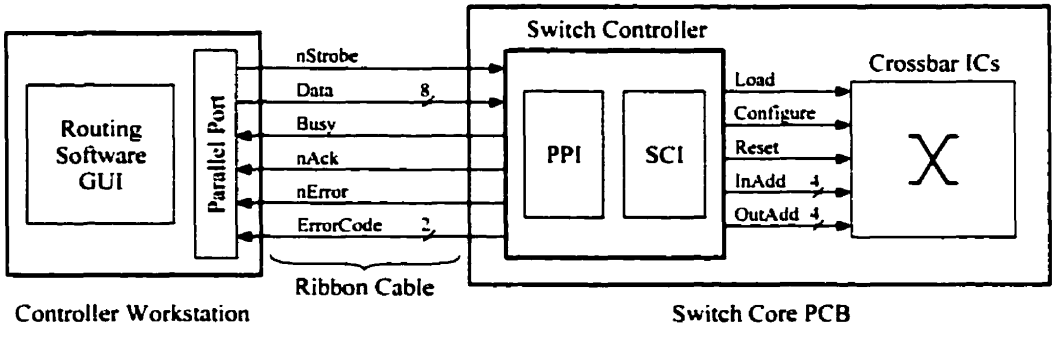


Fig. 4.6. Communication Link between Controller Workstation and Switch Core PCB.

4.2.1. Routing Software on the Controller Workstation

The Routing Software [30] in Fig. 4.7 has an intuitive graphical user interface (GUI) to modify the 16 input-output port mappings of the 16x16 optoelectronic switch core, to save the current switch mapping and to retrieve previously stored mappings. The Routing

Software was developed in collaboration with 3 undergraduate students as part of a term project, supervised by Prof. T. H. Szymanski. An *input-output port mapping* represents a connection from the input port to the output port. The Routing Software is developed using the Visual C++ object-oriented programming environment. The GUI is first constructed by defining dialog controls (buttons) within the dialog box (window). The software execution follows the Event Driven paradigm. When a dialog control is activated (e.g., clicked, selected), the actions associated with the control are performed, which are described using C programming language code.

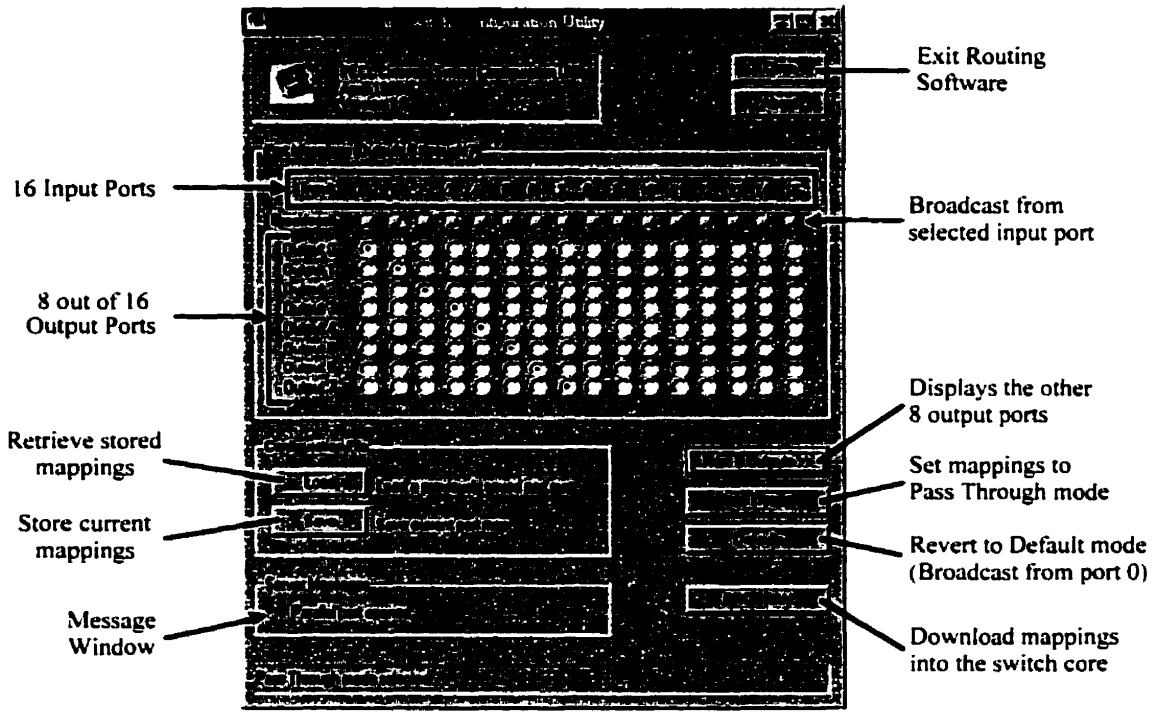


Fig. 4.7. Routing Software Graphical User Interface.

The 16x8 two-dimensional (2D) array of circular radio buttons in the center of Fig. 4.7 displays the mappings for output ports 0 to 7. The mappings for output ports 11 to 15 can be viewed by selecting the "Next 8 Outputs" Button. The programming environment limits each dialog box (window) to display up to 256 dialog controls (buttons). Hence, the mappings are visually organized in two sets of 16x8 array [30].

In Fig. 4.7, each vertical column in the 2D array represents an input port of the switch core. Each horizontal row in the 2D array represents an output port of the switch core. A connection from an input port i to an output port j is established by clicking the button intersecting column i and row j. The connection is displayed as a solid circle and

saved in the Routing Software's internal data structures. A column may have multiple connections, which would correspond to multicasting from an input port. But a row can only contain one connection because an output port cannot be driven by multiple input ports. This restriction is built into the user interface. *Broadcast* from input port i to all 16 output ports can be configured by selecting the square button above column i. The *Pass-Through* configuration, i.e., input port i is connected to output port $j=i$, is shown in Fig. 4.7. This mode is configured by selecting the "Pass Through" Button on the right. The "Default" Button clears the current configuration and reverts to the default mapping of broadcast from input port 0.

The switch core mappings entered into the GUI can be written to a *Configuration File* using the "Save" Button and previously stored Configuration Files can be retrieved using the "Load" Button. The format of the Configuration File is described in Appendix B. When the "Program" Button is selected, the Routing Software configures the switch core using the mappings displayed in the GUI. The transfer procedures and data format are described in Appendix B.

The Message Window reports system messages from the GUI, communication error messages and configuration status messages. Selecting the "Close" Button exits the Routing Software.

4.2.2. FPGA-based Switch Controller

The Switch Controller, adapted from [30], implements the Parallel Port Interface (PPI) and the Switch Configuration Interface (SCI) modules in programmable logic using CMOS FPGA.

Data is transferred between the Controller Workstation and the Switch Core PCB one byte at a time using the parallel port protocol. In Fig. 4.6, the Controller Workstation drives the 8-bit datapath DATA and the data ready signal nSTROBE. The Switch Controller manipulates the signals BUSY, nACK, nERROR and ERRORCODE.

The PPI handles the data transfer and error detection on the 8-bit datapath between the Controller Workstation and the Switch Core PCB over the parallel port ribbon cable. The SCI module receives the routing data from the PPI and configures the crossbar switch ICs, which make up the electrical switch core. Partitioning the Switch Controller into PPI and SCI modules allows the communication and configuration functions to operate in a "pipelined" fashion (i.e., overlapped execution in time). The PPI and SCI modules are Finite State Machines (FSMs). Their operation and implementation in VHDL (Very High Speed Integrated Circuits Hardware Description Language) are presented in Appendix B.

4.2.3. Synthesis Result and Configuration Latency

This section provides the FPGA synthesis results for the Switch Controller, and estimates the time required by the Switch Controller to configure the switch core.

A complete Switch Controller utilizes 25 I/O pins and requires one Altera FLEX 81500 SRAM-based FPGA (part no. EPF81500AGC280-2) [47]. The VHDL design entities are synthesized using the Altera MAX+PLUS II integrated design environment. The implementation contains 11 input pins and 16 output pins, and requires 84 out of 1296 logic cells, which corresponds to a utilization of 6%. Timing analysis by the design tool indicates the critical path is within the PPI's state machine and has a delay of 21.2 ns. Hence, the Switch Controller can be clocked at approx. 47 MHz.

To fully configure the 16x16 optoelectronic switch core, the Switch Controller receives 32 address packets and 1 "configure command" packet from the Routing Software. The transfer of an address packet (input or output) from the Controller Workstation to the Switch Controller requires 4 clock cycles. After an output port address packet is received, the crossbar switch LOAD signal is asserted for 1 clock cycle, at the same time as the next address packet is being downloaded. The processing of a "configure command" packet to load the Configuration Latches requires 4 clock cycles. Therefore, the total configuration takes $(32 \times 4) + 4 = 132$ clock cycles. If the Switch Controller operates at 40 MHz (25 ns clock period), the electrical switch core could be configured in 3300 ns.

4.3. Summary

This chapter proposed the detailed design of the 16x16 optoelectronic switch core, which connects to 16 NICs using optical datalinks. Each switch I/O port has a 8-bit datapath and supports 6.4 Gbit/s data rate per optical datalink. The proposed switch core architecture can handle up to 102.4 Gbit/s switching bandwidth generated by the 16 NICs, and is consistent with the overview design presented in [5].

The packaging of the switch core on a high-speed printed circuit board (PCB) was described. The Switch Core PCB contains 16 Optical Transceiver Modules (OTMs), 8 crossbar switch ICs and a FPGA-based Switch Controller. The switch core is statically configured under software control. The operating principles of these modules were described. Each OTM operates the incoming and outgoing Optobus datalinks to and from a single remote NIC. The independent transmit and receive subsystems perform electrical-to-optical and optical-to-electrical conversions, respectively, on the 8-bit switch core I/O datapaths. The switching function is performed using a bank of 8 static PECL 16x16

crossbar switch ICs in parallel, which can potentially support a peak switching bandwidth of 153.6 Gbit/s. Each crossbar IC operates on 1 bit of the byte-wide datapath at each switch core input port. Internally, the crossbar IC is organized as 16 independent 16-to-1 multiplexers, where each output port is driven by a multiplexer output. By configuring the multiplexer select lines, this structure allows each output port to accept data from any of the 16 input ports, and hence supports broadcast and multicast modes.

The Switch Core PCB connects to a dedicated Controller Workstation over an electrical link. The Routing Software on the workstation communicates with the Switch Controller on the PCB using a full handshake protocol over an 8-bit datapath. The functions of the Routing Software were described. The Routing Software provides an efficient graphical user interface to manipulate the input-output port mappings of the 16x16 switch core. These mappings can be stored in external data files, whose format was described. The Switch Controller implements the communication interface and the control logic to configure the 8 crossbar switch ICs, using field programmable gate array hardware.

The next chapter will describe the NIC and switch core prototypes, constructed according to the functional descriptions provided in this and the previous chapters.

CHAPTER 5

System Prototypes

Prototypes of the Network Interface Card and the Switch Core Printed Circuit Board are constructed for the LAN demonstrator. This chapter summarizes the implementation details of the prototypes, which realize a subset of the complete Fiber Optic LAN design presented in Chapters 3 and 4. The LAN prototypes are constructed using custom PCBs, which are designed for rapid prototyping using FPGAs. The features of this prototyping PCB are given in Section 5.1. The prototype NIC and the prototype optoelectronic Switch Core PCB are described in Sections 5.2 and 5.3, respectively, with circuit diagrams and logical descriptions. Commercial electronic integrated circuits (ICs) are used to implement the modules on the NIC and the Switch Core PCB. Custom IC daughterboards are developed to mount these electronic ICs onto the prototyping PCB, and are described in Section 5.4.

5.1. Prototyping Printed Circuit Board

The NIC and switch core prototypes are constructed on a custom prototyping PCB [31]. The prototyping PCB was developed in-house within the Microelectronics and Computer Systems Laboratory at McGill University by Mr. J Walker and Prof. T. H. Szymanski, and was debugged and prepared by an undergraduate student as part of a summer student training program.

An unpopulated prototyping PCB is shown in Fig. 5.1. This double-sided PCB conforms to the Eurocard 6U x 160 mm standard [61] and measures approx. 233 x 160 mm (9.25 x 6.25 in.). In Fig. 5.1, two 96-pin VME (VERSA-Module Europe) male connectors (DIN 41612 standard) [61] are mounted at the bottom of the PCB, and provide access to power supplies and logic analyzer functions when plugged into a prototyping station.

This PCB is designed for rapid prototyping using the Altera FLEX 81500 FPGA (part no. EPF81500AGC280-2) [47]. The CMOS FPGA has a SRAM-based architecture and contains 1296 logic cells, which is an equivalent logic density of approx. 16,000 gates. The FPGA device has a 280-pin PGA footprint with 208 user I/O pins, 60 power-ground pins and 12 dedicated configuration I/O pins. On the PCB, the FPGA configuration I/O pins are hardwired to a 10-pin connector, which attaches to the ByteBlaster download

cable [62]. Logic functions are programmed into the FPGA by downloading the configuration bits from an external host (such as a PC workstation) running the Altera MAX+PLUS II Design Environment [47].

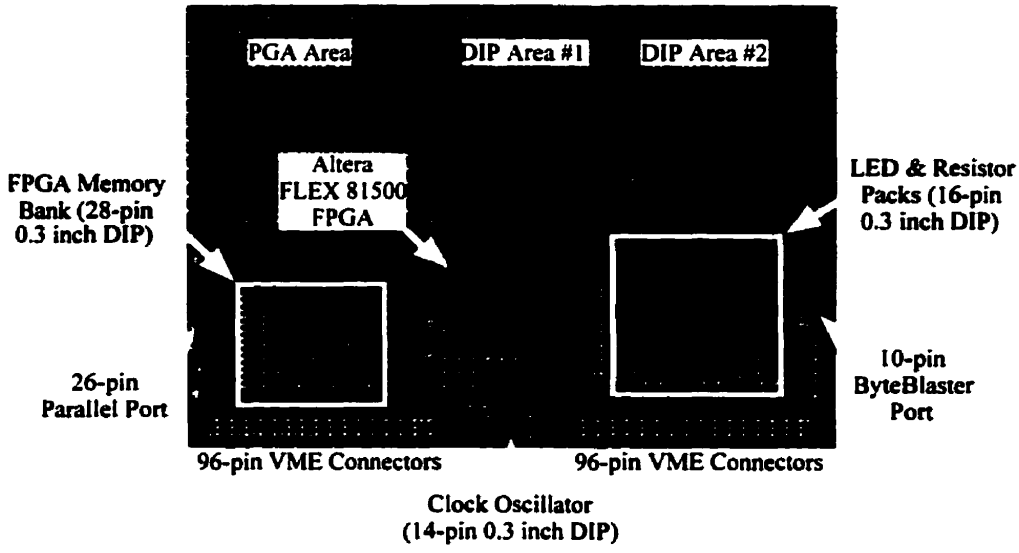


Fig. 5.1. Prototyping PCB used to construct the NIC and switch core prototypes.

Other components hardwired to the FPGA on the PCB include a clock oscillator, external FPGA memory chips, light emitting diodes (LEDs), switches and resistor packs, as shown in Fig. 5.1. For external control and data transfer, a parallel port ribbon cable can be attached to the 26-pin connector on the left side of the PCB. For general purpose prototyping, the PCB features one 26-column by 30-row 0.1 in. pitch Pin Grid Array (PGA) area and two Dual In-line Package (DIP) areas at the top of Fig. 5.1. The DIP areas support DIP packages with 0.3 and 0.4 in. spacings. Using standard wire-wrap pins, ICs can be attached and wired on these areas.

5.2. Prototype Network Interface Card

This section describes a functional prototype NIC, which has been developed to implement a subset of the complete NIC design outlined in Section 3.2. A photograph of the prototype NIC is shown in Fig. 5.2, and contains the following components:

- The Protocol Processor is implemented using the CMOS Altera FLEX 81500 FPGA.
- The Transmitter Module includes one HDMP-1012 transmitter (Tx) IC and three 10H351 CMOS-PECL translators.

- The Receiver Module includes one HDMP-1014 receiver (Rx) IC, three 10E416 PECL signal buffer ICs and three 10H350 PECL-CMOS translators.
- The OTM on the left of Fig. 5.2 includes an Optobus transceiver IC and one 10E416 PECL signal buffer IC.

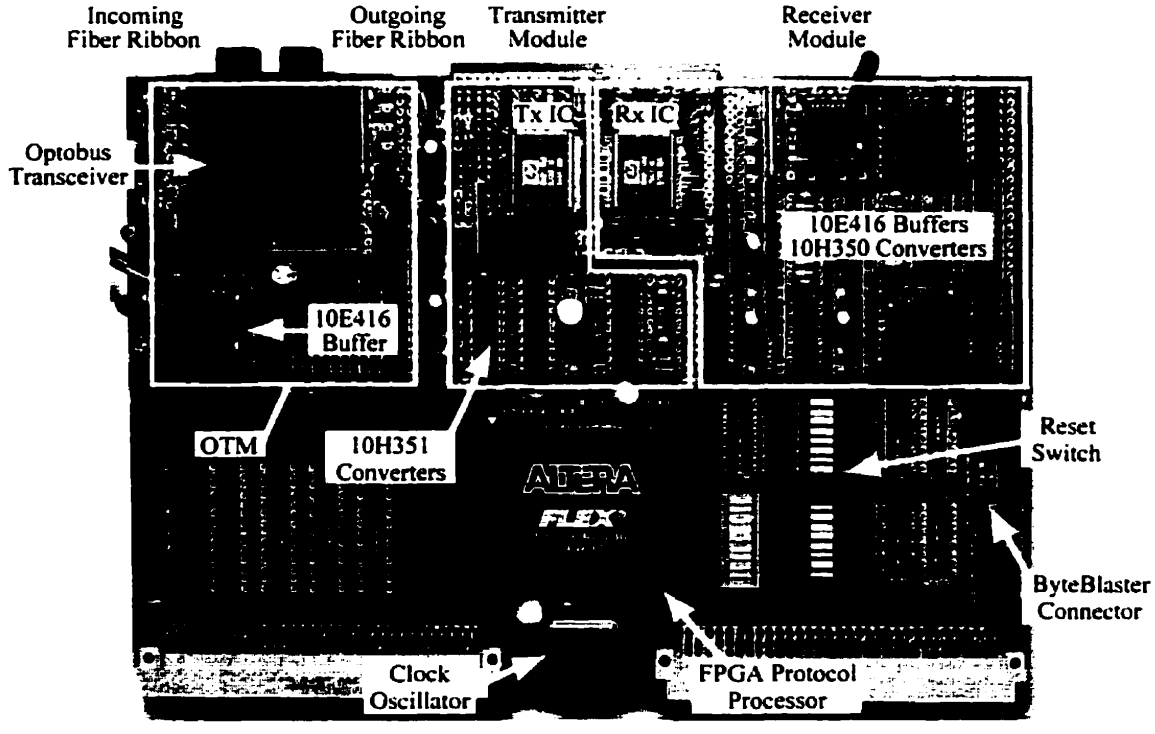


Fig. 5.2. Prototype Network Interface Card.

The transmitter IC, the receiver IC, the Optobus transceiver and the 10E416 buffers are mounted on daughterboards, which are described in Section 5.4. The PCB is powered from a single 5 V supply and all ICs, except the CMOS FPGA, are operated at PECL levels. All PECL signal lines are parallel terminated into 50 Ω using SIP terminating resistor packs, as described in Section 3.1.2. Bypassing and decoupling techniques [10] are applied to the 5 V rail since all PECL levels are referenced to this voltage, as described in Section 5.4. The circuit diagram in Fig. 5.3 highlights the 4 modules on the NIC. The prototype NIC supports 16-bit datapaths between the Protocol Processor, the Transmitter and Receiver Modules.

5.2.1. NIC Transmit Datapath

The transmit function of the NIC is described as follows. The Protocol Processor on the left of Fig. 5.3 is clocked at 12 MHz by an external CMOS clock oscillator on the PCB,

and supplies the frame clock STRBIN, the frame select inputs nCAV and nDAV, and bits D0..3 of the datapath (D4..15 are fixed at logic 0 due to limited board space for CMOS-PECL translators) to the transmitter IC. These transmitter IC I/O ports were described in Section 3.2.2. The Protocol Processor debounces the Reset Switch, which is the initialization input to the transmitter IC signal nRST and the receiver IC signals nSMCRST0..1.

Within the Transmitter Module at the top of Fig. 5.3, the 10H351 translators (Label 1 in Fig. 5.3) convert the single-ended CMOS outputs of the Protocol Processor to differential PECL signals. The uncomplemented PECL outputs of the 10H351 translators (complemented outputs are unused) connect to the single-ended PECL inputs of the transmitter IC, which are terminated to 50 Ω as discussed in Section 3.1.2 (Label 2 in Fig. 5.3). Static PECL logic 0 and 1 are generated by connecting the 10H351 inputs to 5 V and 0 V, respectively. The differential H50 frame clock input STRBIN of the transmitter IC is AC-coupled with 0.1 μF capacitors. The transmitter IC in the prototype is configured to input 16-bit parallel data frames at 12 MHz (DIV1=1, DIV0=0, M20SEL=0) and to generate serial data frames at (16 + 4) bits × 12 MHz = 240 Mbit/s.

The transmitter IC's differential BLL serial data output DOUT is AC-coupled using 0.1 μF capacitors, and passes through the 10E416 signal buffer within the OTM (Label 3 in Fig. 5.3) and connects to the Optobus IC differential input D0. The Optobus transmit subsystem performs E/O conversion and sends the serial data on channel 0 (Ch 0) of the outgoing parallel fiber ribbon to the switch core. The OTM I/O ports were discussed in Section 3.2.4.

5.2.2. NIC Receive Datapath

The receive function of the NIC is described as follows. The Optobus IC within the OTM in Fig. 5.3 receives the 240 Mbit/s serial data from the switch core on channel 0 (Ch 0) of the incoming parallel fiber ribbon (Label 4 in Fig. 5.3). After O/E conversion, the serial data appears on the differential CML output Q0, which is terminated to 50 Ω and is restored to PECL level by the 10E416 signal buffer.

Within the Receiver Module at the bottom of Fig. 5.3, the 10E416 differential output connects to the receiver IC's differential H50 input LIN, which is AC-coupled using 0.1 μF capacitors (Label 5 in Fig. 5.3). The converted frame clock STRBIN from the Protocol Processor differentially drives the receiver IC H50 input DIN.

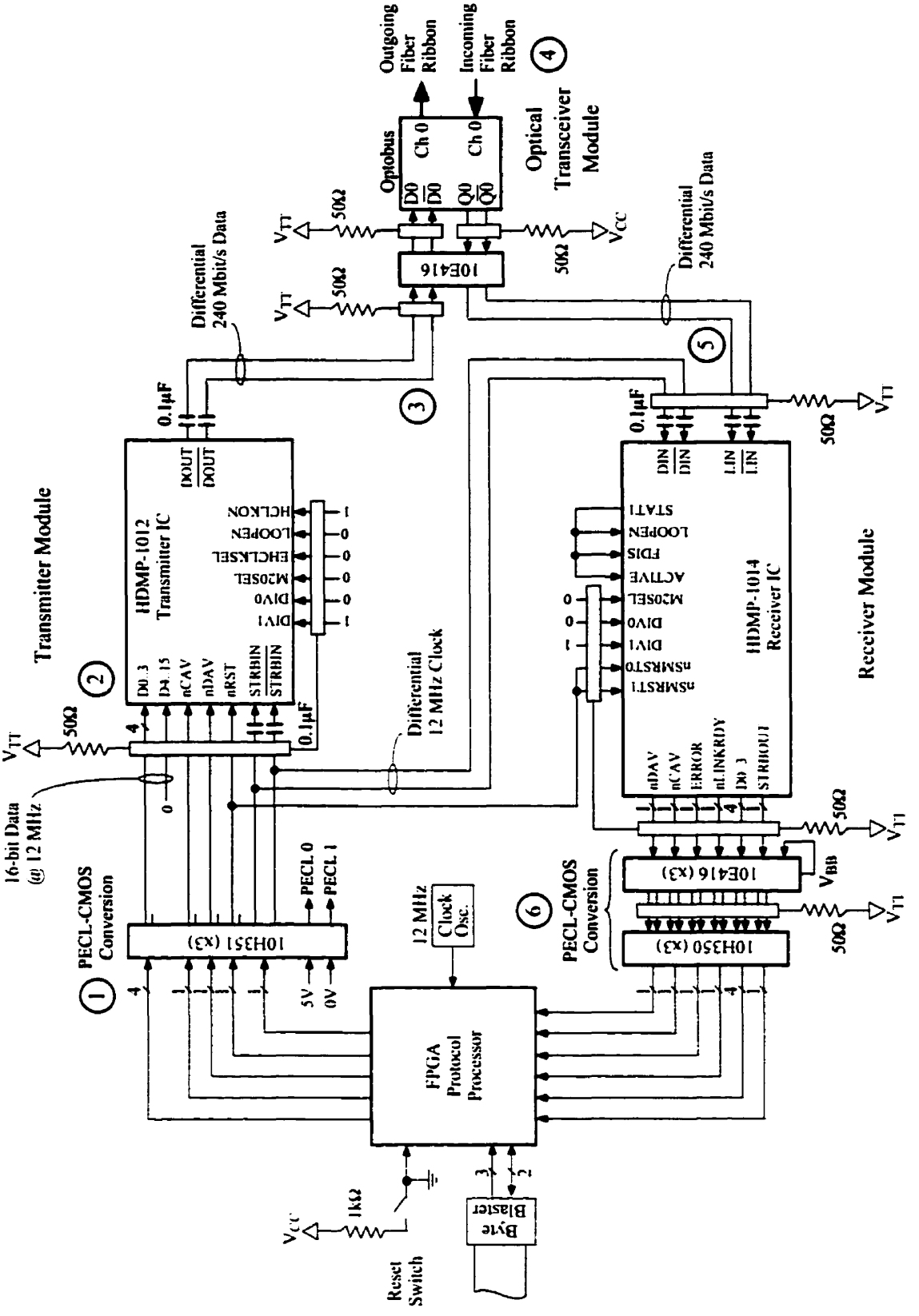


Fig. 5.3. Prototype Network Interface Card Circuit Diagram.

The receiver IC in the prototype is configured to output 16-bit parallel data frames at 12 MHz (DIV1=1, DIV0=0, M20SEL=0). The 16-bit parallel data appears on outputs D0..16, of which D0..3 are of interest (D4..15 are fixed at logic 0 by the transmitter IC). The receiver IC also supplies the recovered frame clock STRBOUT, the frame select outputs nCAV and nDAV, and the link status signals ERROR and nLINKRDY. These single-ended PECL I/O ports of the receiver IC were described in Section 3.2.3.

The 10E416 signal buffers (Label 6 in Fig. 5.3) within the Receiver Module convert the receiver IC single-ended PECL outputs to differential PECL signals. The single-ended signals connect to the 10E416's uncomplemented inputs, while the 10E416's complemented input pins are wired to an internally generated threshold voltage $V_{BB} \approx 3.7$ V [53]. The differential outputs of the 10E416 signal buffers differentially drive the 10H350 PECL-CMOS converters, which provide the single-ended CMOS signals to the Protocol Processor. The Protocol Processor then uses the recovered frame clock STRBOUT to latch the receiver IC outputs.

5.3. Prototype Switch Core Printed Circuit Board

This section describes a functional prototype Switch Core PCB, which has been developed to implement a subset of the switch core architecture outlined in Section 4.1. A photograph of the prototype Switch Core PCB is shown in Fig. 5.4. The switch core supports 1 Input Port and 1 Output Port, and contains the following components:

- The electrical switch core includes one TQ8017 16x16 PECL crossbar IC.
- The single OTM in the center of Fig. 5.4 includes an Optobus transceiver IC and one 10E416 PECL signal buffer IC.
- The Switch Controller is implemented using the CMOS Altera FLEX 81500 FPGA.

The implementation of the prototype Switch Core PCB is similar to that of the prototype NIC. The 16x16 crossbar switch IC, the Optobus transceiver and the 10E416 buffer are mounted on daughterboards, which are described in Section 5.4. The Switch Core PCB is powered from a single 5 V supply and all ICs, except the CMOS FPGA, are operated at PECL levels. All PECL signal lines are parallel terminated into 50 Ω using SIP terminating resistor packs, as described in Section 3.1.2. Bypassing and decoupling techniques [10] are applied to the 5 V rail since all PECL levels are referenced to this voltage, as described in Section 5.4.

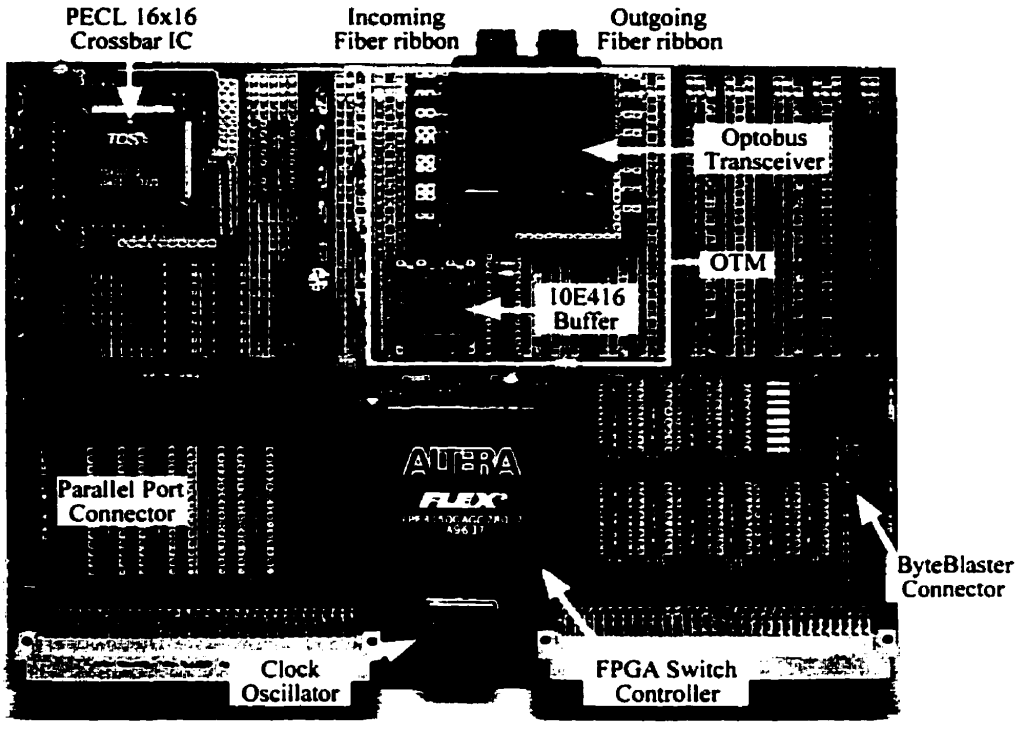


Fig. 5.4. Prototype Switch Core Printed Circuit Board.

The circuit diagram in Fig. 5.5 highlights Input Port 14 and Output Port 15 of the 16x16 optoelectronic switch core. Normally, Input Port OTM i is the same module as Output Port OTM i, where $i \in (0..15)$ (see Fig. 4.1). The Optobus receive subsystem (Label 1 in Fig. 5.5) and the Optobus transmit subsystem (Label 4 in Fig. 5.5) form the same physical module. In the setup of Fig. 5.5, the wiring pattern was modified so that Input Port OTM 14 and Output Port OTM 15 represent the same module. Fig. 5.5 depicts Bit 0 of (byte-wide) Input Port 14 being routed to Output Port 15 through crossbar switch 0.

Within the OTM of Input Port 14 on the left of Fig. 5.5, the incoming parallel fiber ribbon from the prototype NIC connects to the Optobus receive (Rx) subsystem. The 240 Mbit/s serial data is received by the Optobus on channel 0 (Ch 0) of the parallel fiber ribbon (Label 1 in Fig. 5.5) and appears at the differential CML output Q0 after O/E conversion. The CML output is terminated to 50 Ω and is restored to PECL level by the 10E416 buffer within the OTM. The 10E416 buffer differential output connects to input D14 of the crossbar switch 0 (Label 2 in Fig. 5.5).

The OTM of Output Port 15 is on the right of Fig. 5.5. The crossbar switch 0 output Q15 passes through the 10E416 signal buffer within the OTM (Label 3 in Fig. 5.5), and drives the Optobus differential input D0. The Optobus transmit (Tx) subsystem performs

E/O conversion and sends the 240 Mbit/s serial data on channel 0 (Ch 0) of the outgoing parallel fiber ribbon to the prototype NIC (Label 4 in Fig. 5.5).

The FPGA Switch Controller is clocked at 8 MHz by an external CMOS clock oscillator on the PCB (Label 5 in Fig. 5.5). The Parallel Port Interface (PPI) and Switch Configuration Interface (SCI) logic functions are programmed into the FPGA using the ByteBlaster cable (Label 6 in Fig. 5.5). The Switch Controller communicates with the Controller Workstation's Routing Software using the parallel port signals (Label 6 in Fig. 5.5), and provides the CMOS signals to control the crossbar switch IC, as described in Section 4.2.2.

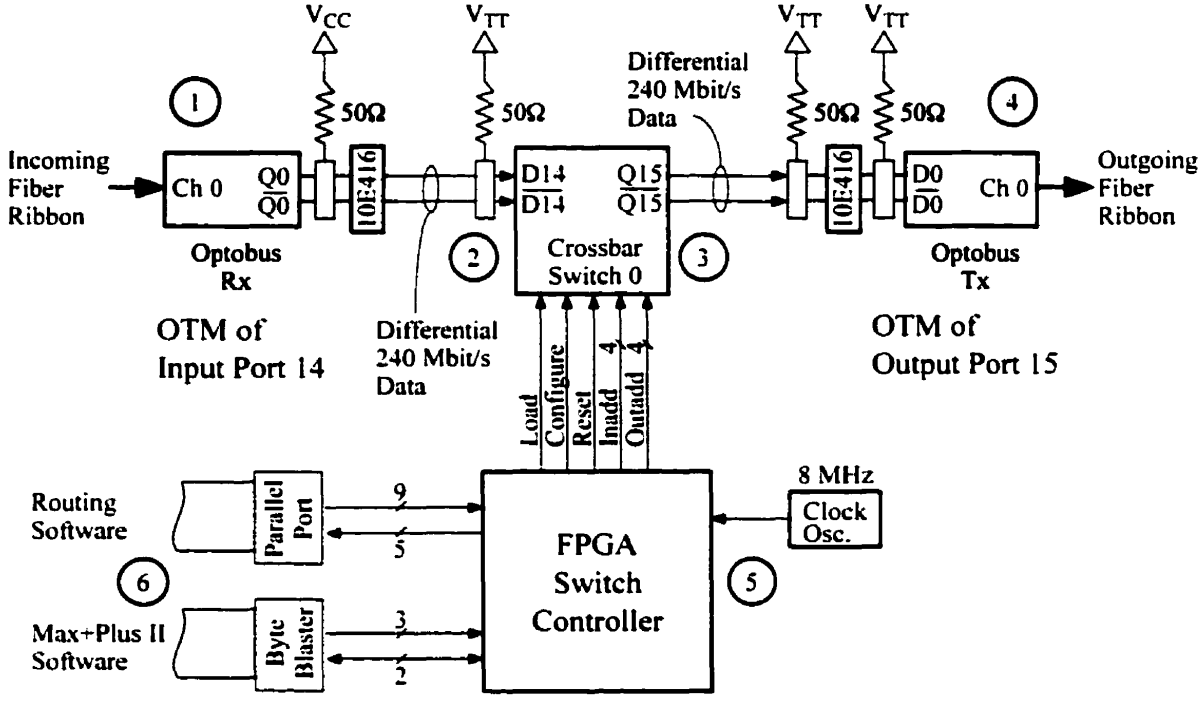


Fig. 5.5. Switch Core Printed Circuit Board Circuit Diagram.

5.4. Integrated Circuit Daughterboards

Commercial off-the-shelf electronic ICs were used to implement the various modules on the prototype NIC in Fig. 5.2 and the prototype switch core in Fig. 5.4. Custom *IC daughterboards* [26] were designed and fabricated using standard 2-sided epoxy glass FR4 material to mount IC devices, without PGA or DIP package footprints, onto the prototyping PCB. These IC devices are soldered onto the surface of the daughterboard and the device I/O are routed, on 0.010 in. PCB traces, to pins [63] mounted under the

daughterboard, which conform to PGA or DIP footprints. Bypass (or decoupling) capacitors and terminating resistors are designed onto the daughterboards, in close proximity to the ICs. PECL signal lines are end terminated using SIP resistors, which have internal 81 Ω pull-up to 5 V and 130 Ω pull-down to 0 V [64]. The Thevenin equivalent of the resistor divider is 50 Ω into 3 V, as described in Section 3.1.2.

High speed switching of device outputs induces high frequency noise on the power supply rails, due to the charging and discharging of capacitive loads [10]. The instantaneous currents generated with switching outputs can cause transient power supply voltage variations. Apart from minimizing system noise on the PCB, maintaining a stable and relatively clean $V_{CC} = 5V$ voltage level is important for PECL systems because the V_{CC} level is used as the reference for the I/O and internal switching bias levels [46]. Any shifts in V_{CC} couple 1-to-1 onto these parameters, hence reducing the noise margins. A *bypass* or *decoupling* capacitor acts as a low impedance supply by storing electrical charge, which is released to the supply rails to counteract the supply voltage variation [58]. At the board level, *bulk* bypass capacitors provide a PCB with enough capacitance to handle long term current demands when many IC outputs switch simultaneously on the PCB [58]. Typically, *electrolytic* capacitors (10-100 μF) are placed at the location where the power supply enters the PCB. These capacitors also filter low frequency (< 1 kHz) noise on the supply rails [10]. All high speed logic ICs require *local* bypass capacitors to supply the instantaneous currents during switching intervals (i.e., when changing states). Local bypass capacitors provide a nearby source of switching currents to prevent degradation in the local power or reference levels, due to the finite inductance of the power distribution network on the PCB [58]. Typically, *monolithic-ceramic* capacitors (0.01-0.1 μF) are placed, as close as possible, across the IC power and ground pins. These capacitors also filter the switching noise generated by active devices, and provide a low inductance path to ground for the return currents during output switching [10].

The following sections describe the daughterboards used in the prototypes, as illustrated in Figs. 5.6 to 5.9. Their corresponding pinout schematics, which map the IC pin numbers to the daughterboard pins, are contained in Appendix C.

5.4.1. Transmitter IC and Receiver IC

Fig. 5.6 shows the daughterboard mounting a pair of Hewlett-Packard HDMP-1012 transmitter IC and HDMP-1014 receiver IC [49], which are used in the Transmitter and Receiver Modules on the NIC, respectively. Both devices have a 80-pin rectangular PQFP (plastic quad flat pack) footprint with 0.80 mm lead pitch. The daughterboard converts the

PQFP footprint to DIP Area footprint and measures approx. 2.7 in. wide by 2.0 in. high. The daughterboard pinout schematic is described in Appendix C.

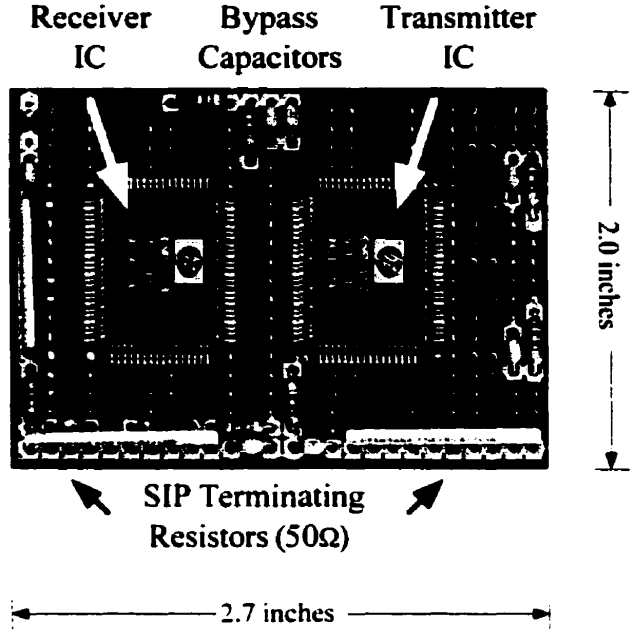


Fig. 5.6. Transmitter and Receiver ICs Daughterboard.

5.4.2. PECL buffer IC

Fig. 5.7 shows the daughterboard mounting the Motorola 10E416 PECL signal buffer IC [53] of the OTM. The 10E416 signal buffer IC has a 28-pin PLCC (plastic leaded chip carrier) footprint, and is secured in a PLCC-to-PGA socket, which is soldered to the daughterboard. The daughterboard mounts the 10E416 signal buffer IC onto the DIP Area and measures approx. 1 in. wide by 1 in. high. The daughterboard pinout schematic is described in Appendix C.

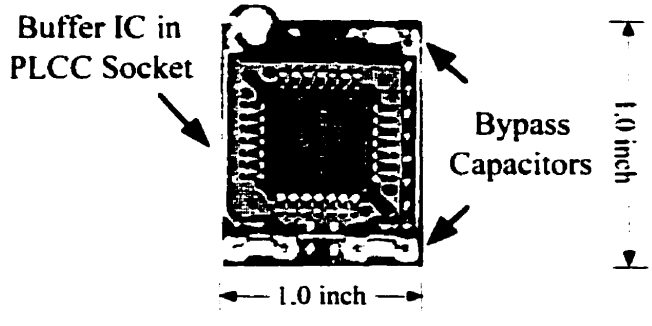


Fig. 5.7. PECL Signal Buffer IC Daughterboard.

5.4.3. Optobus Transceiver IC

Fig. 5.8 shows the daughterboard mounting the Motorola Optobus transceiver IC [65] of the OTM. The Optobus IC has a 196-pin 14x14 PGA footprint with 0.1 in. pin spacings. The daughterboard mounts the Optobus IC onto the DIP Area and measures approx. 2.3 in. wide by 2.0 in. high. The daughterboard pinout schematic is described in Appendix C.

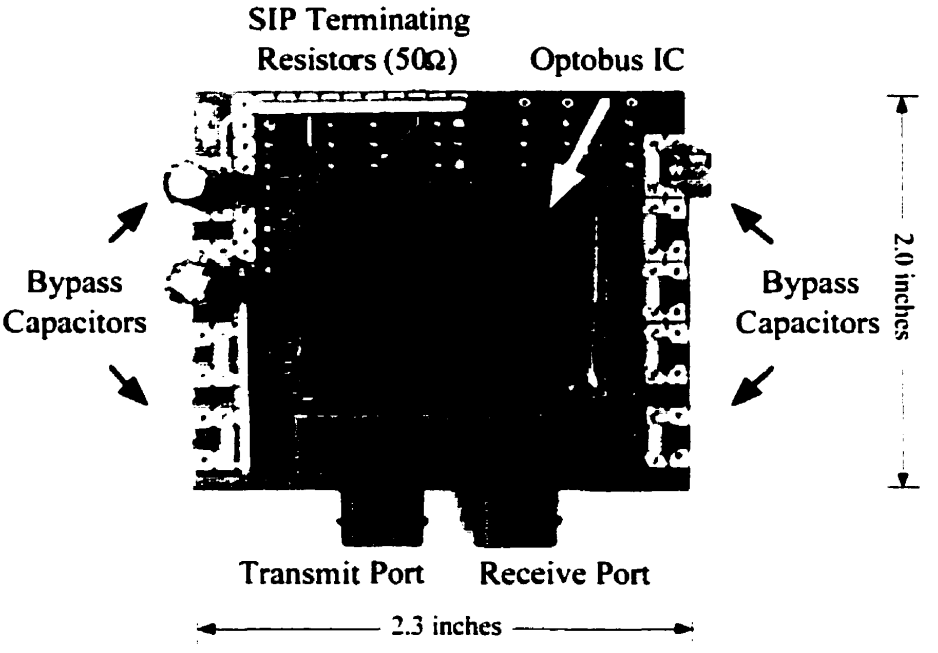


Fig. 5.8. Motorola Optobus Daughterboard.

5.4.4. Crossbar Switch IC

Fig. 5.9 shows the daughterboard mounting the Triquint TQ8017 crossbar switch IC [36] of the Switch Core PCB. The crossbar switch IC has a 132-pin PQFP (plastic quad flat pack) footprint with 0.025 in. lead pitch. The daughterboard mounts the crossbar switch IC onto the PGA Area and measures approx. 2 in. wide by 2 in. high. The daughterboard pinout schematic is described in Appendix C.

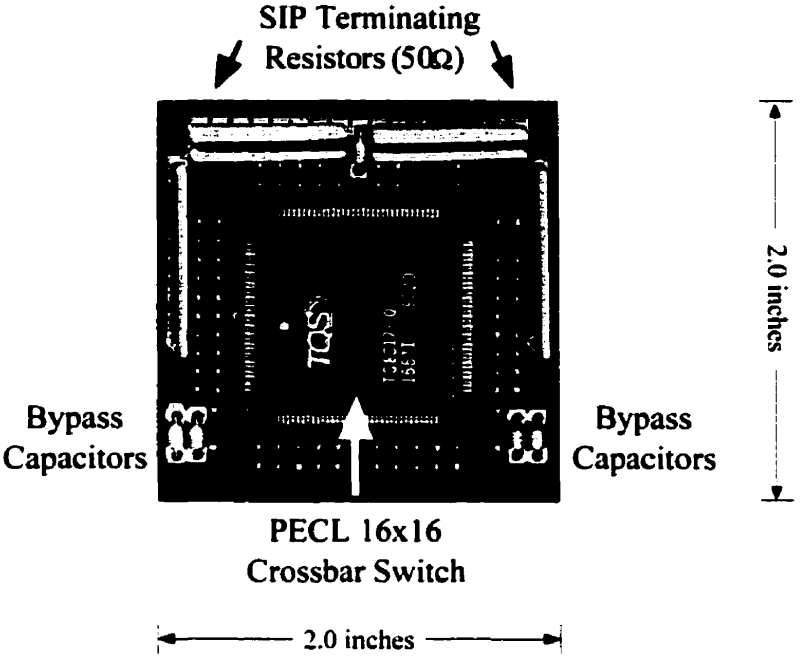


Fig. 5.9. Crossbar Switch IC Daughterboard.

5.5 Summary

In this chapter, functional prototypes of the LAN demonstration system components were presented. The circuit diagrams and logical descriptions of the prototype NIC and the prototype Switch Core PCB were provided. The prototypes are developed from commercial electronic ICs, and are constructed on PCBs customized for prototyping using FPGAs. Daughterboards were developed to mount the electronic ICs, to bypass the IC supplies, and to terminate high speed PECL signal lines.

The next chapter will present the experimental results obtained from the prototypes described in this chapter.

CHAPTER 6

Results, Analysis and Projections

This chapter presents experimental evaluations of the prototype NIC and prototype Switch Core PCB. Section 6.1 demonstrates the transmission of high speed data through the LAN prototypes. Section 6.2 analyzes the packet transmission interval and efficiency. Section 6.3 addresses the scalability issues of the Fiber Optic LAN architecture.

6.1. Packet Flow Measurements

In this section, the transmission of packetized data through the functional prototypes is described and demonstrated. The Fiber Optic LAN demonstration system, consisting of the Switch Core PCB, two 10 m parallel fiber optic datalinks and one NIC, is shown in the photograph of Fig. 6.1.

Fig. 6.2 depicts the transmission of 20-bit CIMT encoded serial frames from one NIC card, through 10 m of parallel fiber ribbon, through the centralized switch core, through 10 m of parallel fiber ribbon, and back to the NIC. For clarity, the prototype NIC is duplicated as a *sending NIC*, which represents the NIC transmit datapath, and as a *receiving NIC*, which represents the receive datapath.

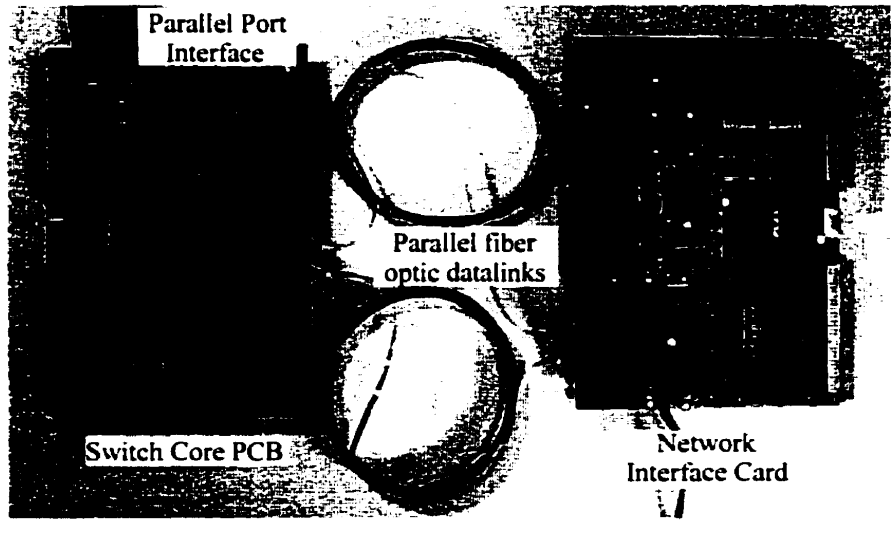
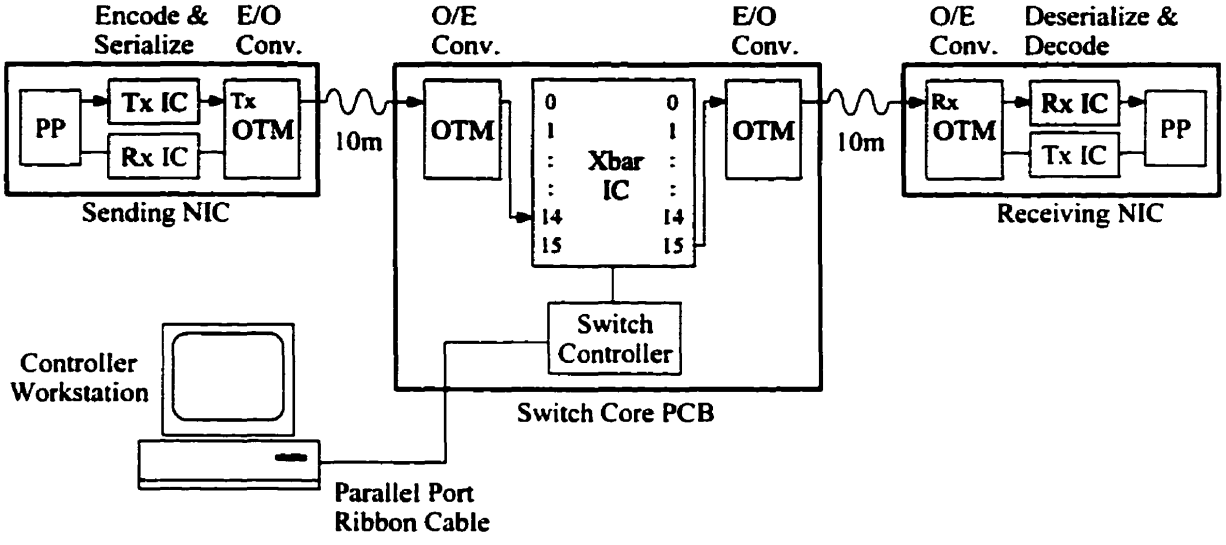


Fig. 6.1. Fiber Optic LAN Components.



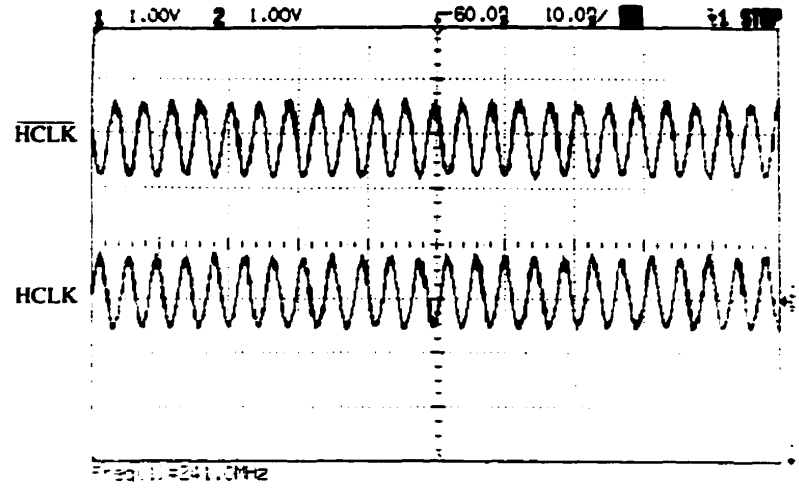
PP is Protocol Processor; Tx is Transmitter; Rx is Receiver; Xbar is Crossbar

Fig. 6.2. Test Setup for LAN Prototypes.

The waveforms in Figs. 6.3 to 6.8 illustrate the packet flow through the network, and were measured using a Hewlett-Packard HP54610B 500 MHz digitizing oscilloscope with 10x 1-MΩ passive probes. The waveform snapshots were captured using a high-resolution CCD hand-held digital camera. In performing the measurements, the probe shield was shorted directly to the circuit board ground, near the signal under test, without using the supplied ground wire. This technique reduces ringing and noise pickup due to the inductive coupling of the long (few inches) ground wire [10].

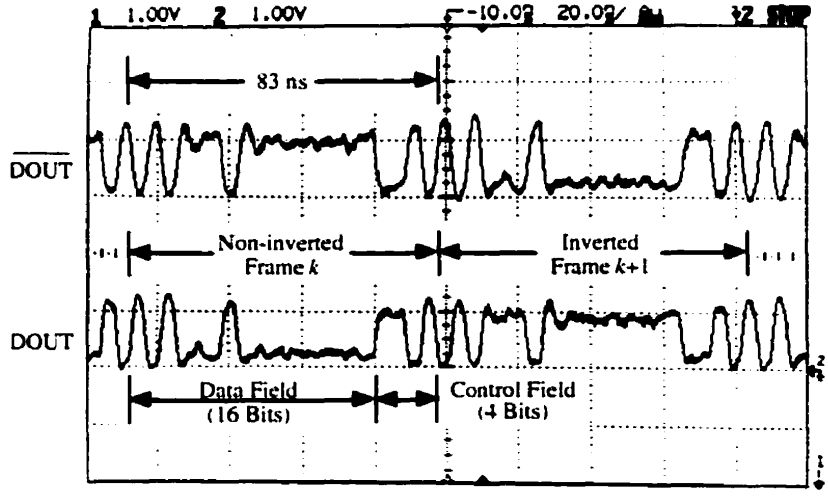
At the sending NIC in Fig. 6.2, the Protocol Processor (PP) transfers 16-bit parallel data frames, with data inputs D0..3 set to "1010", at 12 MHz into the transmitter (Tx) IC. The transmitter IC uses an internally generated 240 MHz clock, shown in Fig. 6.3, to perform serialization. The 16-bit parallel data word "**1010**0000..0" forms the 16-bit Data Frame. The 4-bit Control Field "1101" is appended, as described in Section 3.2.2.3. After CIMT encoding and serialization, the serial data rate at the transmitter IC differential serial data output DOUT is $(16+4)$ bits $\times 12$ MHz $= 240$ Mbit/s, as illustrated in the lower waveform of Fig. 6.4. Each 20-bit serial frame is approx. 83 ns in duration, and the waveform peak-to-peak voltage swing is approx. 800 mV.

In Fig. 6.4, the serial frame k is uninverted, and the following serial frame k+1 is conditionally inverted by the transmitter IC. The 16-bit Data Field and the 4-bit Control Field are highlighted for serial frame k. The observed Data Field contains "10100010..0" because on the transmitter IC we had available, the parallel data input D6 is found to be stuck at logic 1.



Horizontal Scale is 10 ns/division; Vertical Scale is 1 V/division

Fig. 6.3. Transmitter IC High Speed Serial Clock (240 MHz).



Horizontal Scale is 20 ns/division; Vertical Scale is 1 V/division

Fig. 6.4. Serial Frame Inversion on Output of Transmitter IC.

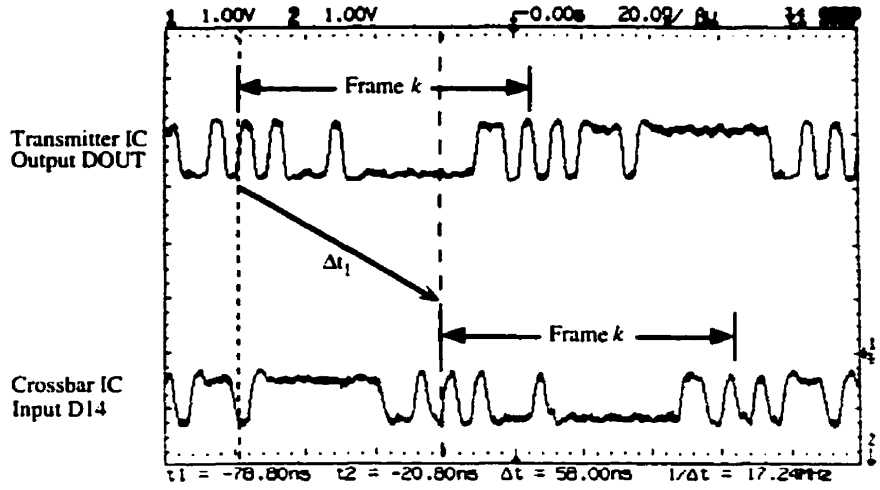
The serial data stream propagates over 10 m of parallel fiber ribbon to the Switch Core PCB. Fig. 6.5 illustrates frame transmission from the transmitter IC output DOUT to the crossbar switch IC input port D14. The observed transmission latency $\Delta t_1 \approx 58$ ns can be broken down as follows:

- propagation delay through sending NIC's OTM is approx. 3 ns.
- fiber delay at 5 ns/m for 10 m of parallel fiber ribbon is approx. 50 ns.
- propagation delay through switch core OTM is approx. 3 ns.

The Routing Software on the Controller Workstation in Fig. 6.2 configures the switch core to connect crossbar switch IC input port D14 to output port Q15. Fig. 6.6 illustrates the input-output propagation delay of the crossbar switch IC, and has an observed latency $\Delta t_2 \approx 3$ ns.

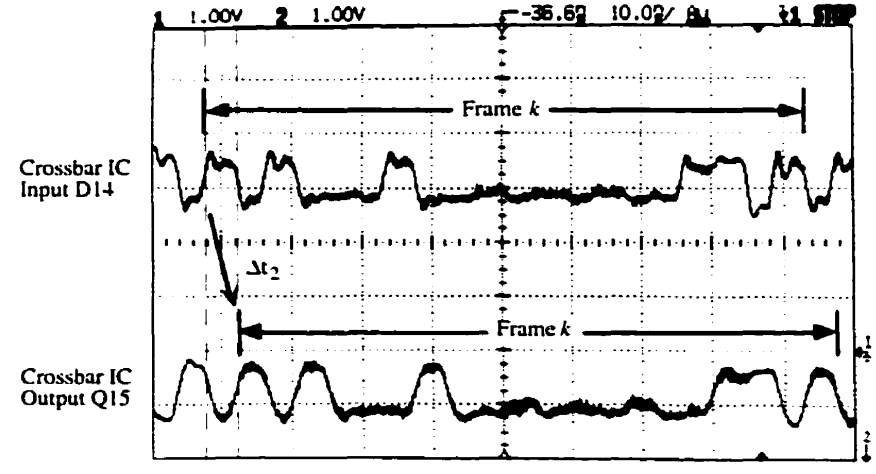
The crossbar switch IC output propagates over 10 m of parallel fiber ribbon to the receiving NIC. Fig. 6.7 illustrates the frame transmission from the crossbar switch IC output port Q15 to the NIC's receiver (Rx) IC input LIN. The observed transmission latency $\Delta t_3 \approx 56$ ns is the sum of the propagation delay through the switch core OTM, the fiber delay through the 10 m parallel fiber ribbon and the propagation delay through the receiving NIC's OTM. At the receiving NIC in Fig. 6.2, the receiver IC deserializes and decodes the data stream back to 16-bit parallel data frames at 12 MHz, which are latched by the Protocol Processor.

The round-trip propagation from the transmitter IC on the sending NIC, over the optical datalink, through the switch core, over the optical datalink and back to the receiver IC on the receiving NIC is demonstrated in Fig. 6.8. The observed transmission latency $\Delta T \approx \Delta t_1 + \Delta t_2 + \Delta t_3 \approx 116$ ns is consistent with the measurements from Figs. 6.5 to 6.7.



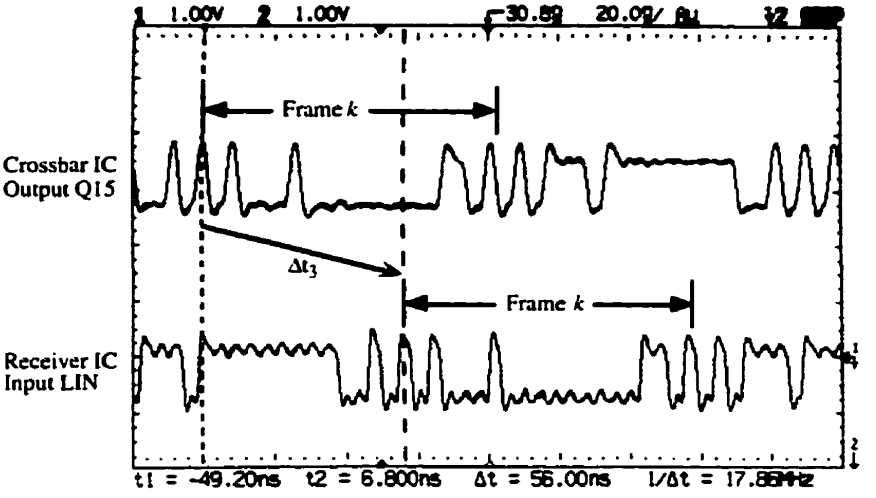
Horizontal Scale is 20 ns/division; Vertical Scale is 1 V/division

Fig. 6.5. Serial Frame Transmission from NIC's Transmitter IC to Crossbar IC, over the Fiber Ribbon with Latency Δt_1.



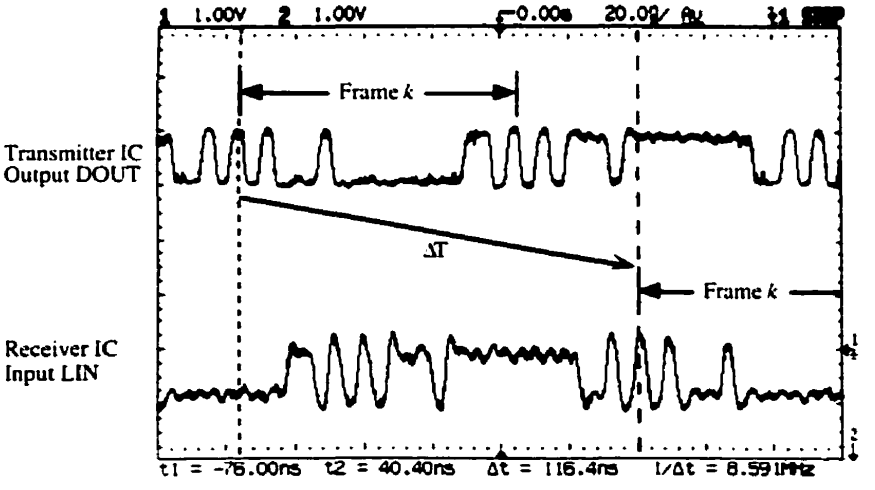
Horizontal Scale is 10 ns/division; Vertical Scale is 1 V/division

Fig. 6.6. Serial Frame Propagation from Input Pin to Output Pin of Crossbar IC with Latency Δt_2.



Horizontal Scale is 20 ns/division; Vertical Scale is 1 V/division

Fig. 6.7. Serial Frame Transmission from Crossbar IC to NIC's Receiver IC, over the Fiber Ribbon with Latency Δt_3.



Horizontal Scale is 20 ns/division; Vertical Scale is 1 V/division

Fig. 6.8. Serial Frame Transmission from the Transmitter IC, through the Crossbar IC, to the Receiver IC with Total Latency $\Delta T=\Delta t_1+\Delta t_2+\Delta t_3$.

6.2. Packet Transmission Analysis

The Switch Controller on the prototype Switch Core PCB described in Chapter 5 is not self-routing. In this section, we describe a simplified packet switching scheme that allows the crossbar ICs within the switch core to be configured dynamically.

Recall from Chapter 2 that the full-scale Fiber Optic LAN contains 16 workstations linked to a centralized 16x16 switch core through their NICs. Each NIC supports 8-bit datapaths over the parallel fiber ribbons to and from the switch core. The Fiber Optic LAN is synchronous, and the time axis is slotted into discrete 20-bit serial *frames*, which are the basic units of transmission. Recall from Section 3.2 that the 128-bit parallel data from the Protocol Processor is partitioned into eight 16-bit data words. Within the Transmitter Module, the 16-bit data words are formatted into 20-bit serial frames by adding 4 control bits. The eight 20-bit serial frames are simultaneously transmitted over 8 dedicated fibers on the parallel fiber ribbon.

In our discussion, we consider a *packet format* with 128 frames per fiber. Fig. 6.9 illustrates the 128-frame packet format, which contains 1 Routing Frame (header) per fiber, 36 Empty Frames per fiber and 91 Data Frames (payload) per fiber. With 16 usable bits per frame, the 91-frame payload could carry approx. 11.6 kbits of user data.

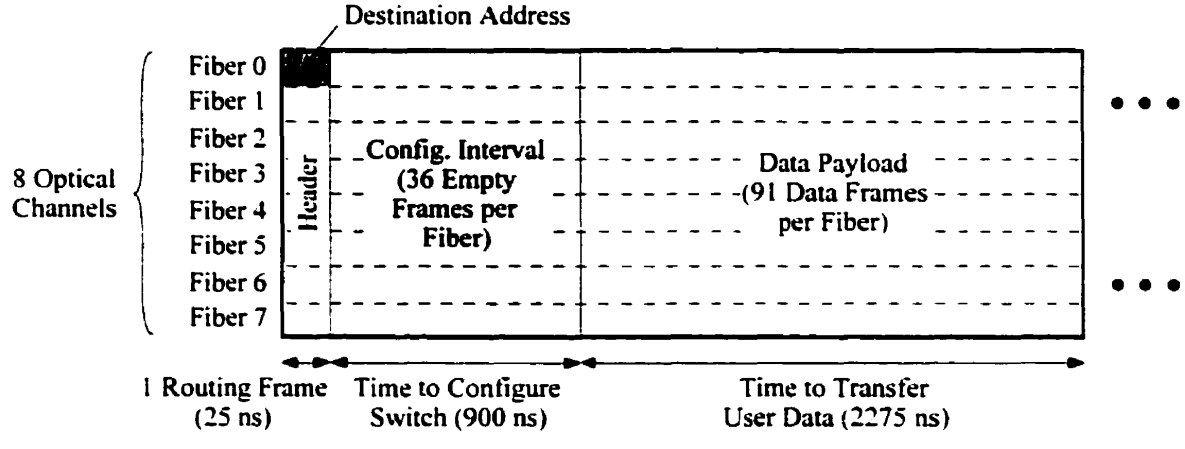


Fig. 6.9. Transmission of a 128-Frame Packet over 8 Fibers.

A packet is sent (over the fiber ribbon) from the sending workstation, is routed through the switch core, and is delivered (over the fiber ribbon) to the receiving workstation within the *Packet Transmission Interval* (PTI). The PTI must allow sufficient time for the Switch Controller to process the routing information within the packet header and to configure the crossbar ICs, before data is sent through the switch

core. The paths between the sending and receiving workstations are being established while the Empty Frames are transmitted, effectively queuing or holding the data at the source until the switches are configured. The analysis of the PTI does not consider output port contention resolution by the switch core, and hence predicts the "best-case" transmission latency within the LAN.

At the start of the switching cycle (i.e., PTI), each of the 16 NICs in the network sends a *Routing Frame* containing the destination NIC address to the switch core, followed by 36 *Empty Frames* (i.e., Fill Frames of Fig. 3.12).

On the Switch Core PCB, the 16 Routing Frames are deserialized by receiver ICs and provide 16 destination NIC addresses in parallel form. Recall from Section 3.2.5 that the receiver IC deserialization requires approx. 60 ns. The centralized switch controller operates at 40 MHz and uses the 16 destination NIC addresses to configure the switch core. The switch controller forms the 16 input-output port mappings. It is estimated that switch configuration would require 33 clock cycles (825 ns), i.e., 2 clock cycles to load each I/O mapping for a total of 32 clock cycles and 1 clock cycle for activation. Therefore, the total switch configuration time is $60 + 825 = 885$ ns.

During header processing and switch configuration, the 36 Empty Frames would maintain link synchronization for 900 ns. Once the switches have been configured, the 91 *Data Frames* can be transferred from the sending NIC through the switch core to the receiving NIC.

The packet transmission interval (PTI) is the time required to transfer the 128-frame packet from the sending NIC to the receiving NIC. Referring to Fig. 6.9, each frame is 20 bits and has 25 ns duration at 800 Mbit/s data rate. The duration of the 128-frame packet in Fig. 6.9 is $(1 + 36 + 91) \times 25$ ns $= 3200$ ns. The serial frames would propagate over 10 m of parallel fiber ribbon at a fiber delay of 5 ns/m. When the fiber delays of 50 ns to travel to and from the NIC are included, the total packet transmission time is 3300 ns. The time used to deliver the data payload is 2275 ns, and therefore the transmission efficiency is approx. 69%.

It is possible to pipeline the switch core to hide the configuration latency. This approach would require limited buffering within the switch core and would raise the efficiency towards 100%. We could also treat each fiber as sending a smaller "linear" packet of 1 Routing Frame, 36 Empty Frames and 91 Data Frames. This choice would allow each fiber to transfer a maximum length Ethernet packet during one PTI.

6.3. Scalability Projections

This section briefly introduces optoelectronic integrated circuit technology and elaborates on the single-chip optoelectronic switching IC proposed in Section 2.2.2. This emerging optoelectronic technology would allow the Fiber Optic LAN architecture to scale to terabit capacities.

Several key components of the Scalable Terabit LAN have been designed and developed. This thesis has presented the detailed designs and the functional prototypes of the workstation NIC and the 16x16 optoelectronic switch core. The workstation NIC can supply 6.4 Gbit/s bandwidth to and from the optoelectronic switch core. The centralized switch core connects to 16 workstation NICs through optical datalinks and supports up to 102.4 Gbit/s switching bandwidth. Operation of the LAN prototypes at 240 MHz has been demonstrated through experimental measurements.

Other major milestones include the fabrication of two field programmable optoelectronic devices in 1995 and 1997. These devices combine optical inputs and outputs with electronic processing abilities, which can be dynamically configured depending on the application. The first chip [66] is a 2x2 mm 0.8 mm CMOS die with approx. 10,000 transistors, 40 electrical I/O and 200 optical I/O. The electrical and optical datapaths operate at several MHz. The second chip [27][28] is a 2x2 mm 0.5 mm CMOS die with approx. 20,000 transistors, 40 electrical I/O and 200 optical I/O. The electrical datapath operates at 200 MHz. The remainder of this section outlines how the components described above can be integrated into an optoelectronic switching system.

6.3.1. Optoelectronic Integrated Circuit Technology

Optoelectronic Integrated Circuit (OEIC) incorporates both optical and electronic devices on a single substrate. This technology brings the benefits of VLSI technology, such as higher performance, increased density and improved reliability, to optical communication systems.

The hybrid integration of high density CMOS electronics with high performance GaAs-based Self Electro-optic Effect Device (SEED) modulators has stimulated much work on photonic switching, optical interconnects and optical computing. In this integration process, arrays of SEED modulators are flip-chip bonded above the CMOS circuits, followed by the removal of the GaAs substrate [23], hence leaving a 2D array of surface-normal optical I/O. Batch fabrication of CMOS-SEED optoelectronic ICs with more than 4,000 optical I/O [67] and yield of 99.95 % has been reported [68]. SEED modulators operate efficiently as optical receivers. But to function as optical transmitters,

an external optical power source is required to "read out" the state of the output modulators. The use of active light sources, e.g., Vertical Cavity Surface Emitting Lasers (VCSELs), as optical transmitters can greatly simplify the optical interface design since they are electrically powered from the OEIC. The integration of multimode VCSEL arrays on CMOS substrates by flip-chip bonding has been demonstrated [24]. This initial CMOS-VCSEL technology is capable of 1.25 Gbit/s data rates per laser at 1 mA threshold current, 1.5 V threshold voltage and 1 mW optical power. The reader should refer to [69] for a review of free-space optical technology, including SEED modulators and VCSEL devices.

Traditional "all-optical" networks have limited data processing abilities. In contrast, optical networks based on optoelectronic technology can process vast amounts of optical data on the OEICs without going off-chip. Furthermore, the direct transfer of data at the chip-level using optics can eliminate the hierarchical interconnect bottleneck typically found in electrical interconnect systems [11]. Often, it is necessary to pass through several levels of electrical interconnections (e.g., chip to board, board to backplane, backplane to cable) to communicate between different parts of the system.

6.3.2. Transition to a Single-Chip Optoelectronic Switch Core

As mentioned in Section 2.2.2, our current "optoelectronic" switch core design implements the switching function with discrete high-speed electrical PECL switches and discrete optical transceivers. The Switch Core PCB of Fig. 4.4 can be collapsed into a single optoelectronic CMOS IC by implementing the functionality of the PCB in the OEIC, as described in [5].

Fig. 6.10 illustrates a conceptual optical setup for such an optoelectronic switching system [21]. The PECL crossbar switch ICs of Fig. 4.4 would be replaced by integrated crossbar circuits on the CMOS substrate. The PCB signal traces, which form the shuffle datapaths between the crossbar switch ICs and OTMs in Fig. 4.4, would be replaced by microelectronic traces. The 16 OTMs in Fig. 4.4 would be replaced by optical receivers (operating in single-ended mode) and VCSEL optical transmitters.

The optical beams would be imaged directly onto the surface-normal optical I/O, which operate at the 850 nm wavelength. The optical I/O are positioned at 62.5 μm and 125 μm spacings. The 16 incoming and 16 outgoing parallel fiber ribbons can be stacked to form a 2D 10x32 fiber array above the OEIC in Fig. 6.10. The horizontal pitch between fibers within the parallel fiber ribbon is 250 μm. Teflon sheets [5] can be inserted between parallel fiber ribbons to maintain a vertical fiber pitch of 500 μm.

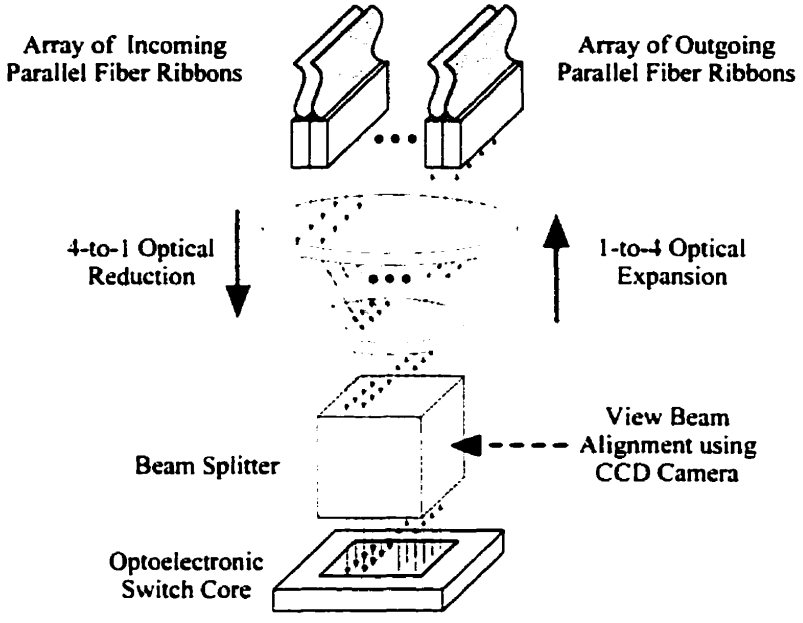


Fig. 6.10. Single Chip Terabit Optoelectronic Switch Core.

A conventional camera lens can be used to focus the 2D array of optical inputs onto the OEIC [5]. With an optical reduction of 4-to-1, the downward beams from the 16 incoming parallel fiber ribbons are focused down to a smaller 2D array that exactly matches the pitch of the SEED optical receivers on the OEIC. In the upward direction, a similar system with an optical expansion of 1-to-4 can be used to couple the 2D array of optical outputs from the VCSEL optical transmitters into the 16 outgoing parallel fiber ribbons to the NIC. Through the beam splitter above the OEIC, a CCD camera mounted horizontally can be used to verify the beam alignment.

Fig. 6.11 illustrates a prototype optoelectronic field programmable logic device, which was fabricated through the 1997/98 Lucent/ARPA/COOP Workshop. The logic functions of this OEIC are user programmable and the design descriptions are reported in [27][28]. The functionality of the Switch Core PCB in Fig. 4.4 can be programmed into this OEIC. In Fig. 6.11, the optical input receivers and output transmitters would replace the 16 OTMs. The integrated crossbar stages would replace the crossbar switch ICs. The VLSI traces (Shuffles 1, 3) would replace the PCB signal traces between the OTMs and the crossbar switch ICs. The 0.5 μm CMOS die measures 2x2 mm, contains approx. 20,000 transistors, 40 electrical I/O pins along the perimeter, and 200 SEED optical I/O on the surface of the die interior.

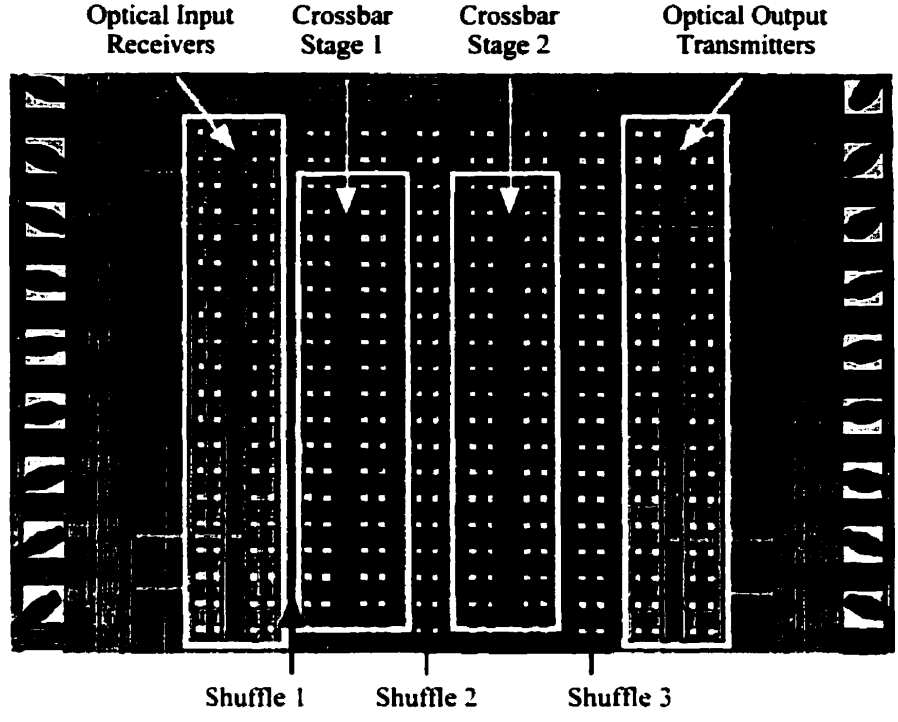


Fig. 6.11. Prototype Optoelectronic Field Programmable Logic Device [27][28].

As described in [5], OEIC technology would allow the fiber optic LAN to scale to Terabits of bandwidth. For example, a 32x32 optoelectronic self-routing switch core with 32 bits per port would require 1024 optical input receivers and 1024 optical output transmitters. Through advanced deep sub-micron IC technology, on-chip clock rates well above 1 GHz will be achievable [70]. If each fiber and optical I/O support data rates of 1.25 Gbit/s, the aggregate optical data bandwidth of the single-chip optoelectronic switch core would be 1.28 Tbit/s. With faster optical I/O clock rates and higher optical I/O densities, bandwidths of 100's of terabits per chip may become available in the future.

6.4. Summary

This chapter demonstrated the operation of the LAN prototypes through experimental measurements, and analyzed packet transmission latency and efficiency. Using optoelectronic integration technology, the Fiber Optic LAN capacities could reach multi-terabit rates, when the functionality of the Switch Core PCB is implemented on a CMOS substrate with integrated optical I/O.

CHAPTER 7

Conclusions

The deployment of gigahertz processors and multi-gigabit network interfaces within the user workstations will propel the necessity for high-bandwidth communication networks. Optical interconnect technologies can provide a vast number of high-speed, high-density interconnections which do not suffer from the limiting physical effects inherent in electrical interconnects. The abundance of optical bandwidth afforded by optical interconnects could spawn novel communication architectures and computing paradigms. These applications would go beyond the conventional use of optical interconnects as functional replacements for electrical interconnects, and would exploit the characteristics unique to the optical domain.

The focus of this thesis has been to develop a system demonstrator based on a scalable terabit Fiber Optic LAN architecture [5][21]. The demonstrator will be used as a testbed for research in high-speed networking technologies, lean protocols and bandwidth-intensive network-oriented applications. The complete system would connect 16 workstations, each having a Network Interface Card and a point-to-point parallel optical datalink to a centralized 16x16 optoelectronic switch core. The switch core would support 102.4 Gbit/s of aggregate bandwidth from the 16 workstations.

Chapter 2 reviewed the Fiber Optic LAN architecture and the novel approaches to achieve multi-gigabit networking. Chapter 3 described the design of the Network Interface Card within the workstations. Chapter 4 dealt with the design of the Switch Core PCB and the Routing Software used to configure the static switch core. Chapter 5 presented the prototypes constructed for the system demonstrator, which implemented a subset of the Fiber Optic LAN. Chapter 6 demonstrated the operation of the prototypes through experimental measurements, and analyzed packet transmission latency and self-routing functions. Finally, the scalability of the LAN architecture to multi-terabit capacities, when the switch core is implemented onto a single optoelectronic IC, is addressed.

As demand for network bandwidth continues to expand at a dramatic rate, monolithic optoelectronic integration technology holds great promise for realizing the speed and performance necessary to satisfy the computing and communication requirements of the future.

Additional work arising from this research include:

- fabricating the single-chip self-routing optoelectronic CMOS switch core proposed in [5][25], using the integrated optical receivers and transmitters discussed in [27][28].
- integrating the optoelectronic switch core into the optical imaging system depicted in Fig. 6.10.
- implementing the "lean" hardware-based network protocols described in Section 2.2.3.
- demonstrating terabit capacity switching over the LAN architecture of Fig. 2.2.

References

[1] Advanced Micro Devices Inc., "AMD Athlon Processor Rockets to 1 GHz", *News Release #20020*, Mar. 6, 2000.
(Available at http://www.amd.com/news/prodpr/20020.html)

[2] Intel Corp., "Intel Introduces Pentium III Processor at One Gigahertz", *Press Release*, Mar. 8, 2000.
(Available at http://www.intel.com/pressroom/archive/releases/sp030800.htm)

[3] A. V. Krishnanmoorthy, J. E. Ford, F. E. Kiamilev, R. G. Rozier, S. Hunsche, K. W. Goosen, B. Tseng, J. A. Walker, J. E. Cunningham, W. Y. Jan, and M. C. Nuss, "The AMOEBA Switch: An Optoelectronic Switch for Multiprocessor Networking Using Dense-WDM", *IEEE Journal of Selected Topics in Quantum Electronics*, Vol. 5, No. 2, Mar.-Apr., 1999, pp. 261-274.

[4] T. E. Anderson, D. E. Culler, and D. A. Patterson, "A Case for NOW (Networks of Workstations)", *IEEE Micro*, Vol. 15, No. 1, Feb. 1995, pp. 54-64.

[5] T. H. Szymanski, A. Au, M. Lafrenière-Roula, V. Tyan, B. Supmonchai, J. Wong, B. Zerrouk, and S. T. Obenaus, "Terabit Optical Local Area Networks for Multiprocessing Systems", *Applied Optics: Information Processing - Special Issue on Massively Parallel Optical Interconnects for Multiprocessor Systems*, Vol. 37, No. 2, Jan. 1998, pp. 264-275.

[6] A. Au, B. Supmonchai, and T. H. Szymanski, "A Fiber Optic Local Area Network Demonstrator", *Proc. Hot Interconnects VII*, Stanford, CA, USA, Aug. 18-20, 1999, pp. 241-245.

[7] IEEE P802.3ae 10Gb/s Ethernet Task Force.
(Available at http://grouper.ieee.org/groups/802/3/ae/index.html)

[8] T. K. Woodward, A. L. Lentine, J. D. Fields, G. Giaretta, and R. Limacher, "A System for Native Ethernet Optical Transport at 10 Gb/s", *Optical Fiber Communication Conference 2000 Technical Digest*, Baltimore, MD, USA, Mar. 5-10, 2000, paper WJ4.

[9] A. Benner, L. Rudolph, E. Schenfeld, T. Sterling, and T. H. Szymanski, "Design Options for Interconnecting a 100+ TFlop/sec Parallel Supercomputer in 2004", Paper and Panel Discussion, *Proc. Fifth International Conference on Massively*

Parallel Processing using Optical Interconnects (MPPOI'98), IEEE Computer Society, Los Alamitos, CA, USA, 1998, pp. 168-173.

[10] H. W. Johnson, and M. Graham, *High-Speed Digital Design: A Handbook of Black Magic*, Prentice-Hall, 1993.

[11] D. A. B. Miller, "Physical Reasons for Optical Interconnection", *International Journal of Optoelectronics*, Vol. 11, No. 3, 1997, pp. 155-168.

[12] D. A. B. Miller, and H. M. Ozaktas, "Limit to the Bit-Rate Capacity of Electrical Interconnects from the Aspect Ratio of the System Architecture", *Journal of Parallel and Distributed Computing*, Vol. 41, No. 1, Feb. 1997, pp. 42-52.

[13] W. J. Dally, and J. Poulton, "Transmitter Equalization for 4-Gbps Signalling", *IEEE Micro*, Vol. 17, No. 1, 1997, pp. 48-56.

[14] K. Chang, W. Ellersick, T.-S. Chuang, S. Sidirospoulos, M. Horowitz, N. McKeown, and M. Izzard, "A 2 Gb/s Asymmetric Serial Link for High-Bandwidth Packet Switches", *Proc. Hot Interconnects V*, 1997, pp. 171-179.

[15] M. Haycock, and R. Mooney, "A 2.5 Gb/s Bidirectional Signaling Technology", *Proc. Hot Interconnects V*, 1997, pp. 149-156.

[16] Y. M. Wong, D. J. Muehlner, C. C. Faudskar, D. B. Buchholz, M. Fishteyn, J. L. Brandner, W. J. Parzygnat, R. A. Morgan, T. Mullally, R. E. Leibenguth, G. D. Guth, M. W. Focht, K. G. Glogovsky, J. L. Zilko, J. V. Gates, P. J. Anthony, B. H. Tyrone, Jr., T. J. Ireland, D. H. Lewis, Jr., D. F. Smith, S. F. Nati, D. K. Lewis, D. L. Rogers, H. A. Aispain, S. M. Gowda, S. G. Walker, Y. H. Kwark, R. J. S. Bates, D. M. Kuchta, and J. D. Crow, "Technology Development of a High Density 32-Channel 16 Gbps Optical Data Link for Optical Interconnect Applications for the Optoelectronic Technology Consortium (OETC)", *IEEE Journal of Lightwave Technology*, Vol. 13, No. 6, Jun. 1995, pp. 995-1016.

[17] D. B. Schwartz, C. K. Y. Chun, B. M. Foley, D. H. Hartman, M. Lebby, H. C. Lee, Chan Long Shieh, Shun Meen Kuo, S. G. Shook, and B. Webb, "A Low Cost, High Performance Optical Interconnect", *IEEE Transactions on Components, Packaging, and Manufacturing Technology - Part B*, Vol. 19, No. 3, Aug. 1996, pp. 532-539.

[18] D. M. Kuchta, J. Crow, P. Pepeljugoski, K. Stawiasz, J. Trewhella, D. Booth, W. Nation, C. DeCusatis, and A. Muszynski, "Low 10 Gigabit/s Optical Interconnects for Parallel Processing", *Proc. Fifth International Conference on Massively Parallel Processing using Optical Interconnects (MPPOI'98)*, 1998, pp. 210-215.

[19] D. Kuhl, K. Drögemüller, J. Blank, M. Ehlert, T. Kraeker, J. Höhn, D. Klix, V. Pickert, L. Melchior, P. Hildebrandt, M. Heinemann, A. Beier, L. Leininger, H.-D. Wolf, T. Wipiejewski, and R. Engel, "PAROLI - A Parallel Optical Link with 15

Gbit/s Throughput in a 12-Channel Wide Interconnection", *Proc. Sixth International Conference on Parallel Interconnects (PI'99)*, 1999, pp. 187-193.

[20] L. Buckman, A. Yuen, K. Giboney, P. Rosenberg, J. Straznicky, K. Wu, and D. Dolfi, "Parallel Optical Interconnects", *Proc. Hot Interconnects VI*, 1998.

[21] A. Au, B. Supmonchai, and T. H. Szymanski, "Testbed for a Scalable Terabit Optical Local Area Network", *Applied Optics: Information Processing*, Vol. 39, No. 23, Aug. 2000, pp. 4131-4142.

[22] T. H. Szymanski, "Parallel Computing with Intelligent Optical Networks", in *Parallel Computing using Optical Interconnections*, eds. K. Li, Y. Pan and S. Q. Zheng, Kluwer Academic Publishing, 1998, pp. 24-46.

[23] K. W. Goossen, J. A. Walker, L. A. D'Asaro, S. P. Hui, B. Tseng, R. Leibenguth, D. Kossives, D. D. Bacon, D. Dahringer, L. M. F. Chirovsky, A. L. Lentine, and D. A. B. Miller, "GaAs MQW Modulators Integrated with Silicon CMOS", *IEEE Photonics Technology Letters*, Vol. 7, No. 4, Apr. 1995, pp. 360-362.

[24] A. V. Krishnamoorthy, L. M. F. Chirovsky, W. S. Hobson, R. E. Leibengath, S. P. Hui, G. J. Zydzik, K. W. Goossen, J. D. Wynn, B. J. Tseng, J. Lopata, J. A. Walker, J. E. Cunningham, and L. A. D'Asaro, "Vertical-cavity surface-emitting lasers flip-chip bonded to gigabit-per-second CMOS circuits", *IEEE Photonics Technology Letters*, Vol. 11, No. 1, Jan. 1999, pp. 128-130.

[25] B. Supmonchai, A. Au, and T. H. Szymanski, "An Optoelectronic CMOS Switch Core for a Terabit Optical Local Area Network", *Proc. Sixth International Conference on Parallel Interconnects (PI'99)*, IEEE Computer Society, Los Alamitos, CA, USA, 1999, pp. 35-42.

[26] A. Au, *Report on Custom Sockets PCB*, Internal Report, Microelectronics and Computer Systems Laboratory, McGill University, Montreal, Canada, Aug. 1997.

[27] T. H. Szymanski, M. Saint-Laurent, V. Tyan, A. Au, and B. Supmonchai, "A Field Programmable Gate Array with Optical I/O", *Proc. Optics in Computing '99*, Optical Society of America, Washington DC, USA, 1999, pp. 149-152.

[28] T. H. Szymanski, M. Saint-Laurent, V. Tyan, A. Au, and B. Supmonchai, "Field Programmable Logic Devices with Optical Input/Output", *Applied Optics: Information Processing - Special Issue on Optics in Computing*, Vol. 39, No. 5, Feb. 2000, pp. 721-732.

[29] E. Laprise, *Transmission of a Video Signal Using Motorola's OPTOBUS Optical Link*, Internal Report, Department of Electrical and Computer Engineering, McGill University, Montreal, Canada, Apr. 1996.

[30] T. Fisher, E. Kucharsky, and P. Thomas, *An FPGA-based Controller for a 16x16 PECL Crossbar Switch*, Internal Report, Department of Electrical and Computer Engineering, McGill University, Montreal, Canada, Apr. 1998.

[31] J.-F. Chamberland, *Rapid Prototyping Board Version 2*, Internal Report, Microelectronics and Computer Systems Laboratory, McGill University, Montreal, Canada, Aug. 1997.

[32] J. Duato, S. Yalmanchili, and L. Ni, *Interconnection Networks: An Engineering Approach*, IEEE Computer Society Press, 1997, pp. 1-34.

[33] A. A. Chien, M. D. Hill, and S. S. Mukherjee, "Design Challenges for High-Performance Network Interfaces", *IEEE Computer*, Vol. 31, No. 11, Nov. 1998, pp. 42-44.

[34] D. B. Schwartz, C. K. Y. Chun, B. M. Foley, D. H. Hartman, M. Lebby, H. C. Lee, Chan Long Shieh, Shun Meen Kuo, S. G. Shook, and B. Webb, "A Low Cost, High Performance Optical Interconnect", *Proc. 45th Electronic Components and Technology Conference (ECTC'95)*, IEEE, New York, NY, USA, 1995, pp. 376-379.

[35] L. B. Aronson, B. E. Lemoff, L. A. Buckman, and D. W. Dolfi, "Low-Cost Multimode WDM for Local Area Networks Up to 10 Gb/s", *IEEE Photonics Technology Letters*, Vol. 10, No. 10, Oct. 1998, pp. 1489-1491.

[36] Triquint Semiconductor, *TQ8017 - 1.2 Gigabit/sec 16x16 Digital PECL Crosspoint Switch*, Beaverton, OR, USA.

[37] D. Bertsekas, and R. Gallager, *Data Networks*, 2nd Ed., Prentice Hall, 1992.

[38] N. J. Boden, D. Cohen, R. E. Felderman, A. E. Kulawik, C. L. Seitz, J. N. Seizovic, and W.-K. Su, "Myrinet: A Gigabit-per-Second Local Area Network", *IEEE Micro*, Vol. 15, No. 1, Feb. 1995, pp. 29-36.

[39] D. Ridge, D. Becker, P. Merkey, and T. Sterling, "Beowulf: Harnessing the Power of Parallelism in a Pile-of-PCs", *Proc. IEEE Aerospace Conference*, 1997, Vol. 2, pp. 79-91.

[40] A. C. Walker, M. P. Y. Desmulliez, M. G. Forbes, S. J. Fancey, G. S. Buller, M. R. Taghizadeh, J. A. B. Dines, C. R. Stanley, G. Pennelli, A. Boyd, J. L. Pearson, P. Horan, D. Byrne, J. Hegarty, S. Eitel, H-P. Gauggel, K-H. Gulden, A. Gauthier, P. Benables, J. L. Gutzwiller, and M. Goetz, "An Optoelectronic Crossbar Switch as a Demonstrator Test-bed for Terabit/s I/O", *Proc. Optics in Computing '99*, Optical Society of America, Washington DC, USA, Apr. 1999, pp. 199-201.

[41] T. M. Slagle, and K. H. Wagner, "Optical Smart-Pixel-based Clos Crossbar Switch", *Applied Optics*, Vol. 36, No. 32, Nov. 1997, pp. 8336-8350.

[42] T. Chaney, J. A. Fingerhut, M. Flucke, and J. S. Turner, "Design of a Gigabit ATM Switch", *Proc. IEEE INFOCOM '97*, Vol. 1, pp. 2-11.

[43] J. W. Lockwood, H. Duan, J. J. Morikuni, S.-M. Kang, S. Akkineni, and R. H. Campbell, "Scalable Optoelectronic ATM Networks: The iPOINT Fully Functional Testbed", *Journal of Lightwave Technology*, Vol. 13, No. 6, Jun. 1995, pp. 1093-1103.

[44] H. Haznedar, *Digital Microelectronics*, The Benjamin/Cummings Publishing Co., 1991, pp. 266-290.

[45] W. R. Blood, Jr., *MECL System Design Handbook*, Fourth Edition, Motorola Inc., 1988.

[46] C. Petty, and T. Pearson, *Application Note AN1406: Designing with PECL (ECL at +5.0V)*, Motorola Inc., Phoenix, AZ, USA, 1992.

[47] Altera Corp., *Altera FPGA Data Book 1996*, San Jose, CA, USA, 1996.

[48] Chu-Sen Yen, R. C. Walker, P. T. Petruno, C. Stout, B. W. H. Lai, and W. J. McFarland, "G-Link: A Chipset for Gigabit-Rate Data Communication", *Hewlett-Packard Journal*, Vol. 43, No. 5, Oct. 1992, pp. 103-116.

[49] Hewlett-Packard Co., *Low Cost Gigabit Rate Transmit / Receive Chip Set Technical Data*, 1994.

[50] Motorola Inc., *MECL Data (DL122/D) Rev. 5*, Phoenix, AZ , USA, Jul. 1993.

[51] P. V. Brennan, *Phase-Locked Loops: Principles and Practice*, MacMillan Press Ltd., 1996, Chapters 1-2.

[52] L. J. Norton, F. Carney, N. Choi, C. K. Y. Chun, R. K. Denton, Jr., D. Diaz, J. Knapp, M. Meyering, C. Ngo, S. Planer, G. Raskin, E. Reyes, J. Sauvageau, D. B. Schwartz, S. G. Shook, J. Yoder and Y. Wen, "OPTOBUS I: A Production Parallel Fiber Optical Interconnect", *Proc. 47th Electronic Components and Technology Conference*, IEEE, New York, NY, USA, 1997, pp. 204-209.

[53] Motorola Inc., *High Performance ECL Data (DL140/D) Rev. 4*, Phoenix, AZ, USA, Jul. 1996.

[54] Motorola Inc., *PC94DL0400 Optobus I Multichannel Optical Data Link*, Phoenix, AZ, USA, 1996.

[55] J. D. Crow, and A. F. Benner, "Intramachine Communications", in *Handbook of Fiber Optic Data Communications*, eds. C. DeCusatis, E. Maass, D. P. Clement, and R. C. Lasky, Academic Press, 1998, pp. 331-384.

[56] T. Satake, T. Arikawa, P. W. Blubaugh, C. Persons, and T. Uchida, "MT multifiber connectors and new applications", *Proc. 44th Electronic Components and Technology Conference (ECTC'94)*, 1994, pp. 994-999.

[57] J. Grula, *Application Note AN1572: Applying the Optobus I Multichannel Optical Data Link to High-Performance Communication Systems: SCI, Fibre Channel, and ATM*, Motorola Inc., Phoenix, AZ, USA, 1996.

[58] J. E. Buchanan, *Signal and Power Integrity in Digital Systems*, McGraw-Hill, 1996, pp. 123-172.

[59] W. J. Dally, and J. W. Poulton, *Digital Systems Engineering*, Cambridge University Press, 1998, pp. 25-78.

[60] C. Peacock, *Interfacing the Standard Parallel Port*, Feb. 1998. (Available at http://www.beyondlogic.org/spp/parallel.htm)

[61] Vero Electronics Ltd., *Product Catalog '97*, Chandlers Ford, Hampshire, U.K., p. 8.26.

[62] Altera Corp., *ByteBlaster Parallel Port Download Cable Data Sheet Version 2*, San Jose, CA, USA, Jan. 1998.

[63] Advanced Interconnections Corp., *Single In-line Adapters Type 138*, Catalog #14, West Warwick, RI, USA, 1996, p. 47.

[64] CTS Corp., *Single-In-Line Conformal Series 770*, CTS Resistor Products Catalog, Berne, IN, USA, p. 15.

[65] Motorola Inc., *Optobus Data Sheet*, Logic Integrated Circuits Division, Chandler, AZ, USA, 1995.

[66] S. S. Sherif, S. K. Griebel, A. Au, D. Hui, T. H. Szymanski, and H. S. Hinton, "Field-Programmable Smart-Pixel Arrays: Design, VLSI Implementation, and Applications", *Applied Optics*, Vol. 38, No. 5, Feb. 1999, pp. 838-846.

[67] A. L. Lentine, K. W. Goossen, J. A. Walker, L. M. F. Chirovsky, L. A. D'Asaro, S. P. Hui, B. T. Tseng, R. E. Leibenguth, J. E. Cunningham, W. Y. Jan, J. M. Kuo, D. Dahringer, D. Kossives, D. D. Bacon, G. Livescu, R. L. Morrison, R. A. Novotny, and D. B. Buchholz, "Optoelectronic VLSI Switching Chip with greater than 4000 Optical I/O based on Flip Chip Bonding of GaAs-AlGaAs MQW Modulators and Detectors to Silicon CMOS", *Conference on Lasers and Electrooptics '96*, Optical Society of America,Washington DC, USA, 1996, pp. 517-518.

[68] K. W. Goossen, "Optoelectronics/VLSI", *Proc. 48th IEEE Electronic Components and Technology Conference*, 1998, pp. 771-777.

[69] V. Tyan, "Design of an Electrical-Optical Interface for a 307.2 Gb/s Free-space Optical Backplane", *M.Eng. Thesis*, Department of Electrical and Computer Engineering, McGill University, Oct. 1998, Chapter 2.

[70] Semiconductor Industry Association, *The National Technology Roadmap for Semiconductors*, San Jose, CA, USA, 1997.

APPENDIX A

Network Interface Card

This appendix deals with the following topics:

- Architectural overview of the Altera Field Programmable Gate Array (FPGA).
- HDMP-1012 transmitter IC electrical I/O summary and timing characteristics.
- HDMP-1014 receiver IC electrical I/O summary and timing characteristics.
- Detailed description of the receiver IC state machine controller (SMC).

A.1. Protocol Processor

This section presents the architecture of the Field Programmable Gate Array (FPGA), which is used to implement the Protocol Processor on the prototype Network Interface Card, and the Switch Controller on the Switch Core Printed Circuit Board.

A.1.1. Altera Field Programmable Gate Array

An FPGA is a device with an array of uncommitted logic resources, which can be generically configured and interconnected. Altera's Flexible Logic Element Matrix (FLEX) FPGA architecture [47], shown in Fig. A.1, incorporates basic building blocks called *Logic Elements* (LE). Eight LEs are grouped together to form a *Logic Array Block* (LAB). Signal interconnections between the LABs are provided by a series of fast, continuous channels that run the width and height of the device. The *Input-Output Element* (IOE) at the ends of the interconnects interface to a bi-directional device I/O pin, for a total of 204 pins. The FLEX 81500 device contains 1296 LEs grouped into 162 LABs, which are arranged into 6 rows and 27 columns [47]. The logic and interconnections are configured with CMOS SRAM elements.

A *row interconnect* contains 216 horizontal channels and terminate into 8 IOEs at each end. Each row IOE can drive an input signal onto at most 2 channels within its row interconnect, and can select 1 of 27 row interconnect channels as an output signal. A *column interconnect* contains 16 vertical channels and terminate into 2 IOEs at each end. Each column IOE can drive an input signal onto at most 2 channels within its column interconnect, and can select 1 of 8 column interconnect channels as an output signal.

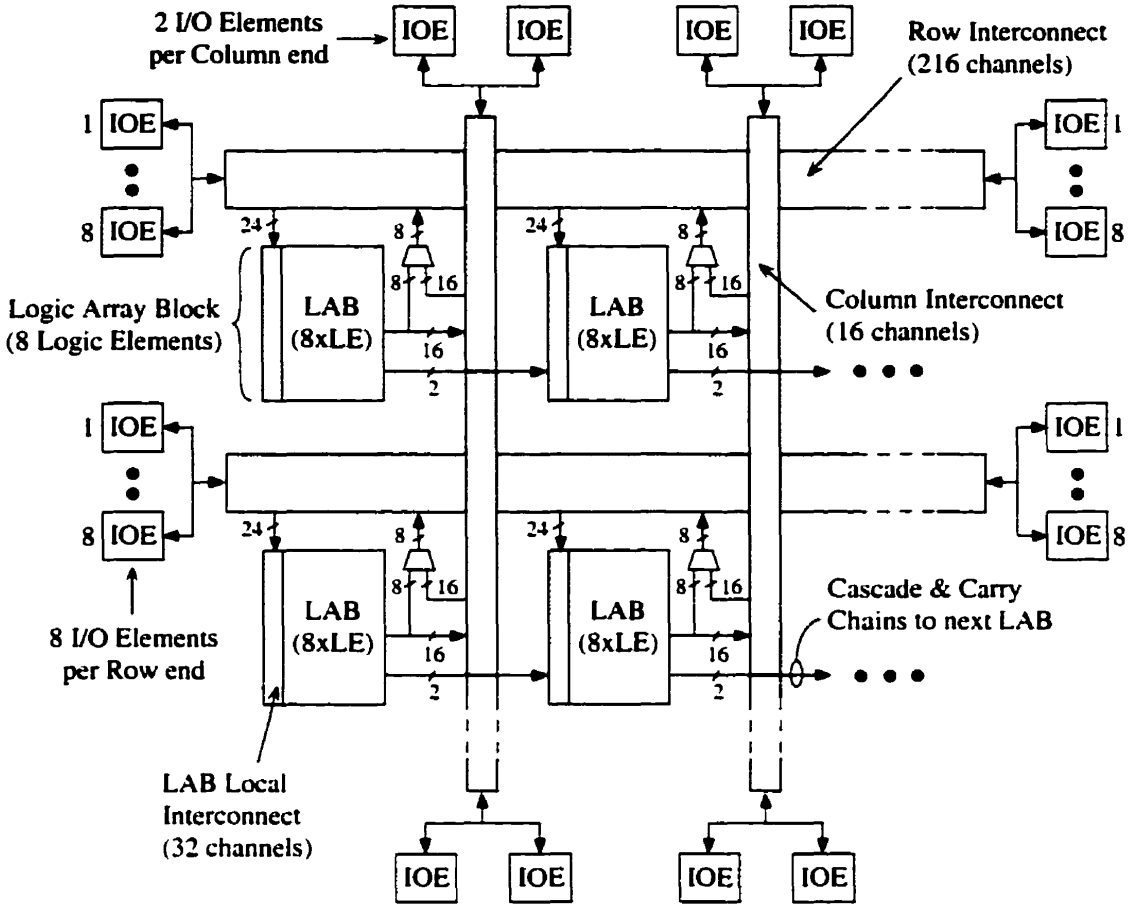


Fig. A.1. Flexible Logic Element Matrix (FLEX) FPGA Architecture [47].

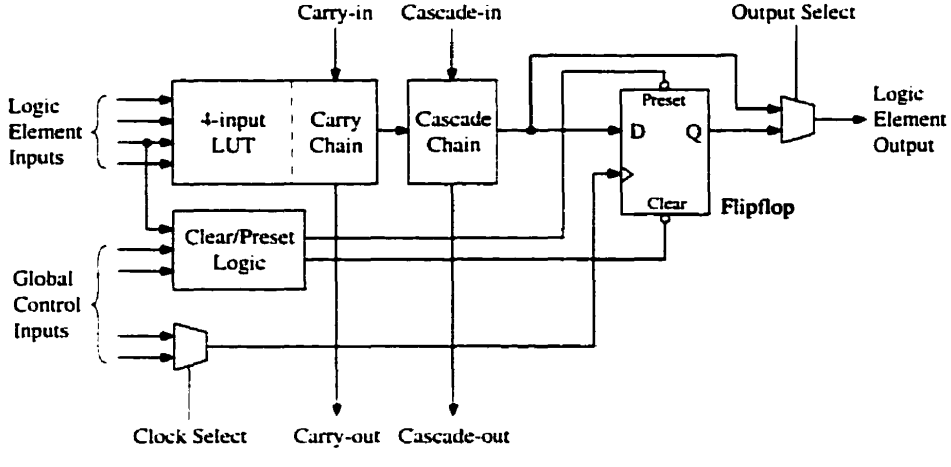


Fig. A.2. Logic Element Functional Diagram [47].

In Fig. A.1, the 32-channel LAB *local interconnect* forms the LAB inputs (i.e., 4 channels per LE) from 24 row interconnect channels and the output feedback of 8 local LEs. The LAB outputs can drive up to 16 column interconnect channels (i.e., up to 2 channels per LE) and 8 row interconnect channels (i.e., 1 channel per LE). The LAB access to the row interconnect is shared with the column interconnect via multiplexing.

The functional diagram of a LE [47] is shown in Fig. A.2. Each LE contains a 4-input look-up table (LUT), a programmable flipflop, and carry chain and cascade chain logic. The *LUT* can compute the boolean function of 4 variables. The *programmable flipflop* can be configured for D, T, JK, or SR operation, and can be bypassed for purely combinational functions.

The clock, preset and clear signals on the flipflop can be driven by global signals from dedicated input pins, which support low-skew, device-wide signal distribution. All LEs within the same LAB use the same global control signals.

The *carry* and *cascade chains* are dedicated high-speed datapaths that connect adjacent LEs within the LAB, and all LABs in the same row. The carry chain provides very fast carry-forward functions for adders and counters. The cascade chain implements wide-input functions. Adjacent LUTs can be used to compute portions of the function in parallel, and the cascade chain serially combines the intermediate values.

A.2. Transmitter Module

This section describes in detail the electrical I/O and timing characteristics of the transmitter IC, which is used to implement the Transmitter Module on the prototype Network Interface Card.

A.2.1. Transmitter IC Electrical I/O Summary

The transmitter IC contains circuitry for frame assembly, parallel-to-serial conversion and clock generation, as shown in Fig. A.3. The transmitter IC electrical I/O are summarized in Table A.1.

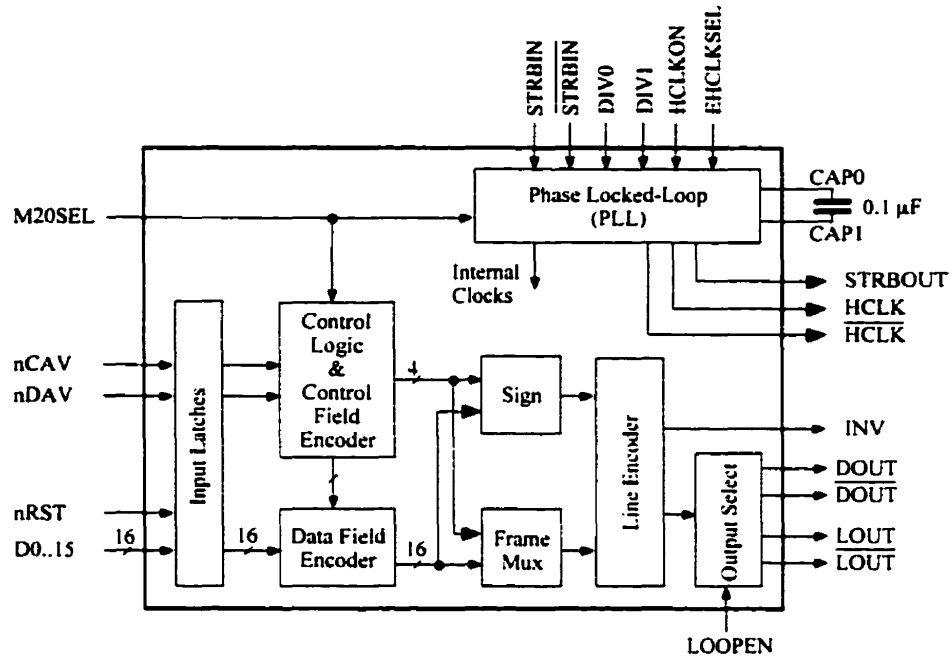


Fig. A.3. HDMP-1012 Gigabit Transmitter IC Block Diagram.

Port Name	Port Type	Description
CAP0, CAP1	Capacitor	External loop filter capacitor connections (0.1 µF)
nCAV	PECL input	Control frame select (Active low)
D0..D16	PECL input	16-bit parallel data input
nDAV	PECL input	Data frame select (Active low)
DIV0, DIV1	PECL input	Operating range select (Refer to Table 3.1, Chapter 3)
DOUT, $\overline{\text{DOUT}}$	BLL output	Serial data output
EHCLKSEL	PECL input	External serial clock input enable (Active high)
HCLK, $\overline{\text{HCLK}}$	BLL output	Serial clock output
HCLKON	PECL input	Serial clock output enable (Active high)
INV	PECL output	Conditional inversion status (Active high)
LOOPEN	PECL input	Serial data output select: 0 selects DOUT, 1 selects LOUT
LOUT, $\overline{\text{LOUT}}$	BLL output	Loopback serial data output
M20SEL	PECL input	Parallel data size select: 0 selects 16-bit mode, 1 selects 20-bit mode
nRST	PECL input	Initialize transmitter IC (Active low)
STRBIN, $\overline{\text{STRBIN}}$	H50 input	Parallel frame clock input
STRBOUT	PECL output	Retimed parallel frame clock output

Table A.1. Summary of HDMP-1012 Gigabit Transmitter IC I/O Ports.

A.2.2. Transmitter IC Timing Characteristics

The timing characteristics of the serialization of a parallel 16-bit data frame k is illustrated in Fig. A.4. The rising edge of the 40 MHz frame clock STRBIN is used as the timing reference.

The parallel data inputs D0..15 and the control inputs nDAV, nCAV and nRST are latched on the rising edge of the frame clock STRBIN. These input signals must satisfy the setup time $t_{setup} > 6$ ns and hold time $t_{hold} > 0$ ns. The retimed frame clock STRBOUT appears after $t_{strbout} = 1.5$ ns, with a maximum delay of 3 ns.

From Fig. A.4, the latency from the input of the parallel data D0..15 to the output of the serial frame on DOUT is $t_{dout} = (2 \times \text{serial bit duration}) - 0.5$ ns. At an encoded serial bit rate of 800 Mbit/s, the serial bit duration is 1.25 ns and the latency $t_{dout} = 2.0$ ns.

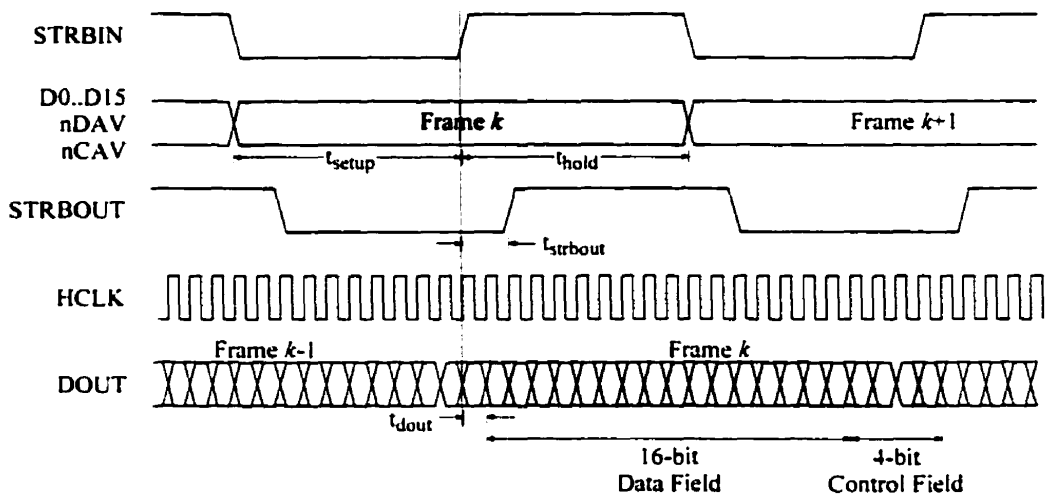


Fig. A.4. Transmitter IC Serialization Timing Diagram.

A.3. Receiver Module

This section describes in detail the electrical I/O and timing characteristics of the receiver IC, which is used to implement the Receiver Module on the prototype Network Interface Card. The State Machine Controller within the receiver IC is also discussed.

A.3.1. Receiver IC Electrical I/O Summary

The receiver IC contains circuitry for clock recovery and generation, serial-to-parallel conversion, and data and link status extraction, as shown in Fig. A.5. The receiver IC electrical I/O are summarized in Table A.2.

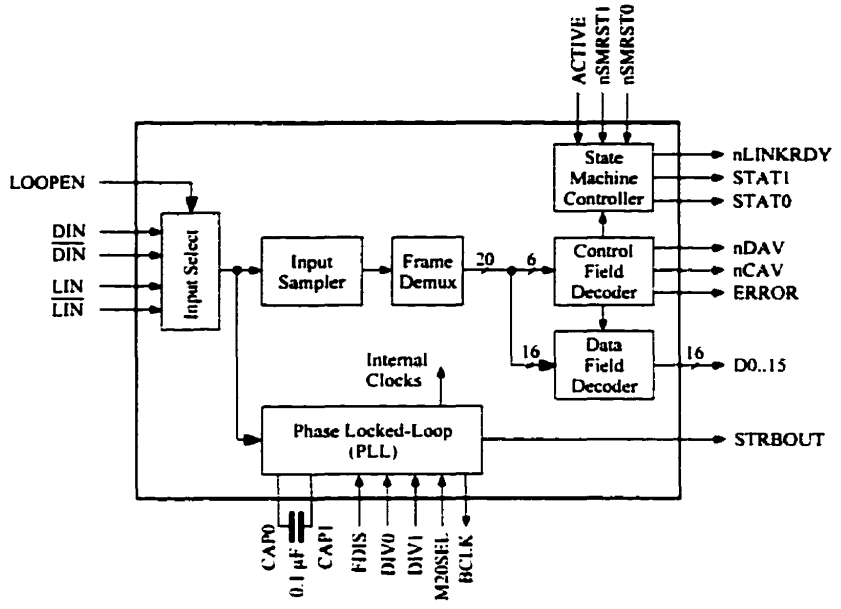


Fig. A.5. HDMP-1014 Gigabit Receiver IC Block Diagram.

Port Name	Port Type	Description
ACTIVE	PECL input	Chip enable (Active high) Normally driven by state machine
BCLK, $\overline{\text{BCLK}}$	BLL output	Recovered serial clock output
CAP0, CAP1	Capacitor	External loop filter capacitor connections (0.1 µF)
nCAV	PECL output	Control frame available (Active low)
D0..D16	PECL output	16-bit parallel data output
nDAV	PECL output	Data frame available (Active low)
DIN, $\overline{\text{DIN}}$	H50 input	Serial data input
DIV0, DIV1	PECL input	Operating range select (Refer to Table 3.1, Chapter 3)
ERROR	PECL output	Frame Error (Active high)
FDIS	PECL input	Frequency detector disable (Active high) Normally driven by state machine
LIN, $\overline{\text{LIN}}$	H50 input	Loopback serial data input
LOOPEN	PECL input	Serial data input select: 0 selects DIN, 1 selects LIN Normally driven by state machine
nLINKRDY	PECL output	Valid data or control frames (Active low)
M20SEL	PECL input	Parallel data size select: 0 selects 16-bit mode, 1 selects 20-bit mode
nSMRST0, nSMRST1	PECL input	State machine reset inputs (Active low)
STAT0, STAT1	PECL output	State machine status outputs
STRBOUT	PECL output	Recovered parallel frame clock output

Table A.2. Summary of HDMP-1014 Gigabit Receiver IC I/O Ports.

A.3.2. Link Startup and Synchronization

In Fig. A.5, the receiver IC contains a State Machine Controller (SMC), which is used to perform the functions of link startup, frame synchronization and error checking. The Conditional Invert with Master Transition (CIMT) line code provides a guaranteed transition at a fixed, defined location in every frame. During link startup, the PLL determines the frequency of the incoming serial data and locates the Master Transition, which is then used to establish and to monitor frame synchronization.

The SMC monitors the lock conditions, and controls the Input Select unit and the PLL through its outputs STAT0 and STAT1. These output signals denote the current status of the SMC, and are used to control the modules within the receiver IC. Status output STAT1 is externally connected to control input signals LOOPEN, FDIS and ACTIVE, whereas status output STAT0 is unused. By providing the SMC status outputs, the receiver IC can be configured externally to operate in one of the four supported link configurations [49].

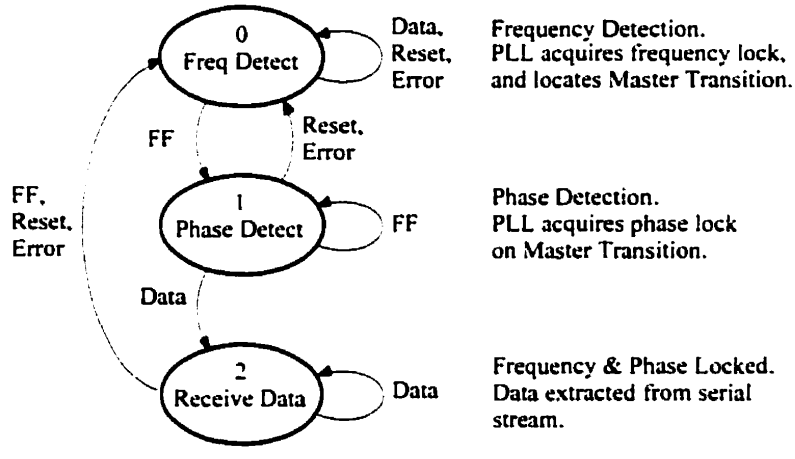


Fig. A.6. State Machine Controller State Diagram.

The state diagram of the SMC is shown in Fig. A.6. The SMC goes into state 0 (frequency detect state) when the receiver IC is reset (i.e., reset input nSMRST0=0 or nSMRST1=0) or when errors are detected (output ERROR=1). In state 0, status output STAT1=0, which forces the receiver IC to use DIN as the serial input, and enables frequency acquisition on the local 40 MHz reference clock by the PLL. The frequency detection circuit operates only on simple square-wave Fill Frames. The periodic reference clock signal emulates a Fill Frame. The rising edge of the reference clock (i.e., Master Transition of the Fill Frame) is used to establish an unambiguous frame reference.

Upon frequency lock, the SMC goes into state 1 (phase detect state) and sets status

output STAT1=1. The Frequency-Phase Detectors in the PLL switch from frequency to phase detection mode. At the same time, the receiver IC switches the serial input from the reference clock on input DIN to the data stream on input LIN.

The sample of the Master Transition is used as an indication of the phase error. The 2 bits forming the Master Transition (i.e., Control Field bits $CF_{1..2}$) are monitored to detect a locked condition within an accepted phase error of ±22.5°. If bits CF_1 and CF_2 are not complementary for 2 or more consecutive frames, it is considered a frame error and a link startup is initiated. In such a case, the SMC returns to state 0, forcing STAT1=0, which switches the input back to the reference clock. This process is repeated until the Master Transition is found and an error-free condition exists.

When frequency and phase locks are achieved, the SMC proceeds into state 2 (data receive state), where the receiver IC decodes the serial input and outputs valid data. The Master Transition of the Data or Control Frames is then used to maintain frame lock and alignment. Typical lock times will be on the order of several milliseconds, which can be improved by pulling the reference oscillator slightly off frequency. Lock times under 200 μs are achievable by adding phase modulation or programmed delay in the reference oscillator path [48].

A.3.3. Receiver IC Timing Characteristics

Fig. A.7 illustrates the deserialization timing of the receiver IC, with respect to a 20-bit serial frame *k*. The falling edge of the recovered frame clock STRBOUT is always aligned to the serial data frame's boundary, while the rising edge is in the center of the data frame. The rising edge of STRBOUT can thus be used by the Protocol Processor to latch the decoded parallel data frame and status information.

There is a latency of 2 serial frames (or equivalently 2 frame clock periods) from the input of the serial frame to the output of the parallel frame D0..D15 and associated status signals. The synchronous parallel data outputs (D0..D15) and status outputs (nDAV, nCAV, ERROR, nLINKRDY) appear with a typical delay of $t_{d1} \approx 2$ ns [49] from the falling edge of STRBOUT. The SMC outputs, STAT0 and STAT1, are updated with a typical delay of $t_{d2} \approx 4$ ns [49] after the falling edge of STRBOUT.

Let T_{data} denote the parallel data output latency and $T_{strbout}$ denote the parallel frame clock period. The parallel data output latency is approximated as $T_{data} = 2T_{strbout} + t_{d1}$. For 40 MHz parallel frame clock, $T_{strbout} = 25$ ns, and thus the latency $T_{data} \approx 52$ ns.

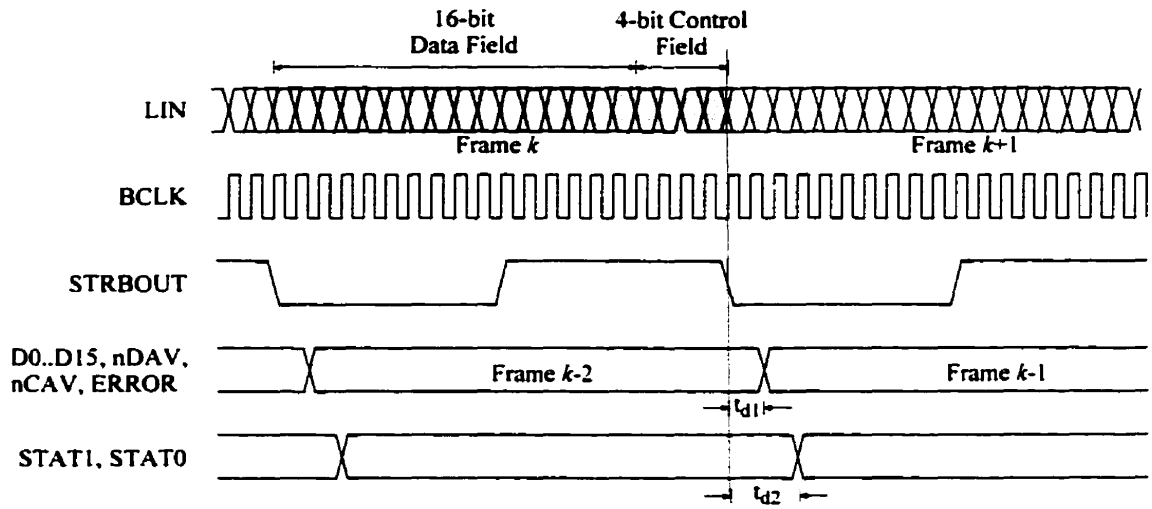


Fig. A.7. Receiver IC Deserialization Timing Diagram.

APPENDIX B

Optoelectronic Switch Core

This appendix deals with the following topics:

- Timing characteristics and configuration procedures of the crossbar switch IC.
- Routing Software configuration file format.
- Detailed description of the FPGA-based Switch Controller.
- Overview of the communication protocol used to transfer data between the Routing Software on the Controller Workstation and the Switch Controller on the Switch Core PCB.
- Detailed description of the Parallel Port Interface (PPI) and the Switch Configuration Interface (SCI) within the Switch Controller.
- VHDL source code for the Switch Controller, the PPI and the SCI.

B.1. Crossbar Switch IC

This section describes the timing characteristics and configuration procedures of the crossbar switch IC, which implements the switch core in Chapter 4.

B.1.1. Timing Characteristics and Configuration Procedures

The timing diagram [36] is shown in Fig. B.1. To initiate switch configuration, an input-output port mapping is placed on INADD3..0 and OUTADD3..0, with setup time $t_{setup} \geq 0$ ns and hold time $t_{hold} \geq 3$ ns, relative to the transitions of signal LOAD, as illustrated in Fig. B.1. The 4-bit input port address is written into the Program Latch specified by the output port address, while LOAD=1 for duration $t_{pulse} \geq 7$ ns. This step is repeated for the other output ports as necessary. The Program Latches can be configured in any sequence and an arbitrary number of times.

As a final step, after all the Program Latches are loaded with the desired input port addresses, signal CONFIGURE is set to 1 for duration $t_{pulse} \geq 7$ ns. All 16 Configuration Latches simultaneously load the output of their Program Latches and drive the multiplexer select lines. Typically, the propagation delay from an input port to an output port is $t_{d1} < 2$ ns. After configuration, the output data is valid within $t_{d2} \leq 5$ ns after the rising edge of CONFIGURE. Clearing the internal latches with RESET=1 (pulse width >

10 ns) forces all 16 output ports to select input port D0.

From Fig. B.1, the minimum time required to configure the 16x16 crossbar switch IC is as follows. The loading of each input port address into the Program Latches requires 10 ns ($t_{pulse} + t_{hold}$). The loading of the Configuration Latches requires 7 ns (t_{pulse}). Therefore, switch configuration can be completed in $(10\text{ns} \times 16 \text{ ports}) + 7\text{ns} = 167$ ns.

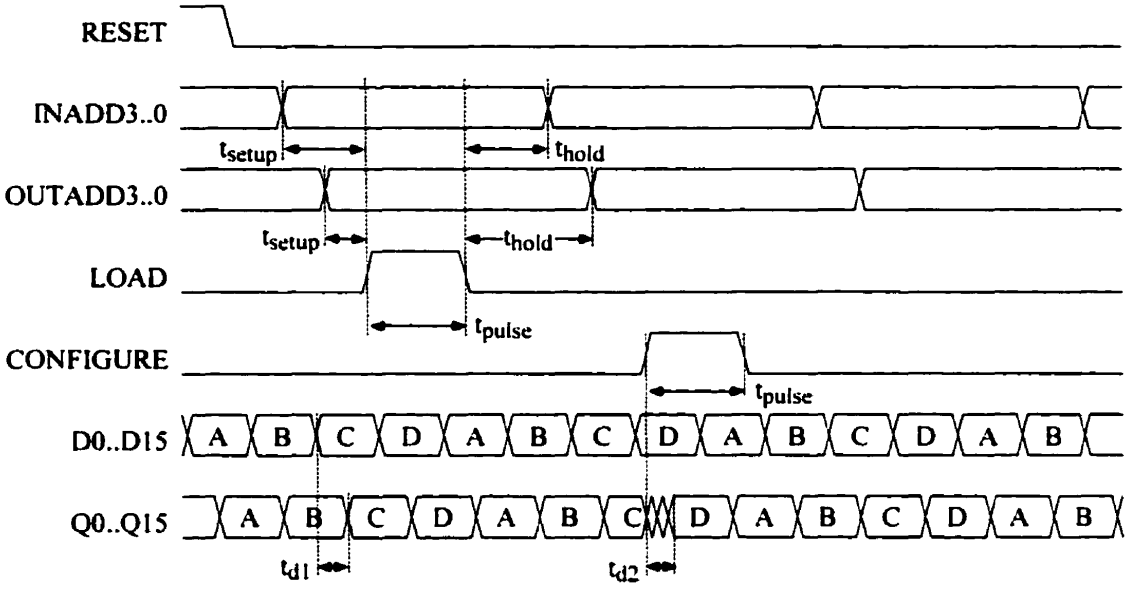


Fig. B.1. Crossbar Switch IC Timing Diagram.

B.2. Routing Software on the Controller Workstation

This section discusses the Configuration File used to store switch core mappings, and describes the procedure and data format used by the Routing Software to communicate with the Switch Controller on the Switch Core PCB.

B.2.1. Configuration File Format

The switch core mappings entered into the Routing Software GUI [30] can be written to a *Configuration File* using the "Save" Button, and previously stored Configuration Files can be retrieved using the "Load" Button. The Configuration File is a plain text file and an example is shown in Fig. B.2. Lines starting with the number sign (#) are ignored by the Routing Software and may be used for comments. During saving, the Routing Software inserts the filename, the time and date of creation at the beginning of the data file. The next section always contain 2 columns of 16 numbers, where the first

column represents the output port *j* and the second column represents the input port *i*. Each line is unambiguously interpreted as output port *j* is connected to input port *i*, and is derived from the 2D array by recording, for each row *j*, the column position *i* of the single connection (i.e., solid circle).

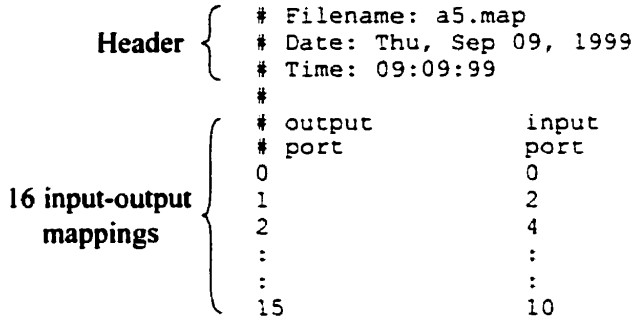


Fig. B.2. Sample Configuration File.

B.2.2. Transfer Procedures and Data Format

When the "Program" Button is selected, the Routing Software initiates the switch core configuration procedure using the mappings displayed in the GUI. For each output port, the Routing Software generates the two 4-bit port addresses in the input-output port mapping, starting with output port 0. The port addresses are formatted into 8-bit packets, whose format is shown in Fig. B.3. An *Even Parity* bit is computed on bits 6 through 0 and is stored in bit 7, the most significant bit (MSB). The *Address* field in bits 3..0 contains the 4-bit input port address or the 4-bit output address. The 3-bit *OpCode* field in bits 6..4 identifies the Address field as an input port (001) or output port (010).

0 0 1 Input port address
0 1 0 Output port address
0 1 1 Configure switch (for all 16 mappings)
1 0 0 Reset switch

Fig. B.3. Data Transfer Packet Format.

The Routing Software downloads the input-output port mappings to the Switch Controller, starting from output port 0 through port 15. In each input-output port mapping transfer, the input port address packet is sent first, followed by the output port

address packet, because this sequence is imposed by the Switch Controller. The address packets are not stored in the Switch Controller. Therefore, each input-output port mapping received by the Switch Controller is immediately forwarded to the crossbar ICs. When all 16 mappings (i.e., 32 address packets) have been transferred, the Routing Software sends a "configure command" packet (with OpCode field set to 011) to the Switch Controller, to activate the new routing configuration for the entire switch core. In total, the Routing Software transmits 33 packets to the Switch Controller to configure the 16x16 switch core.

B.3. FPGA-based Switch Controller

The Switch Controller, adapted from [30], implements the Parallel Port Interface (PPI) and the Switch Controller Interface (SCI) modules in programmable logic using CMOS FPGA. Fig. B.4 shows the block diagram of the Switch Controller, which corresponds to the top-level VHDL entity *switch_control*. The Switch Controller I/O ports are summarized in Table B.1.

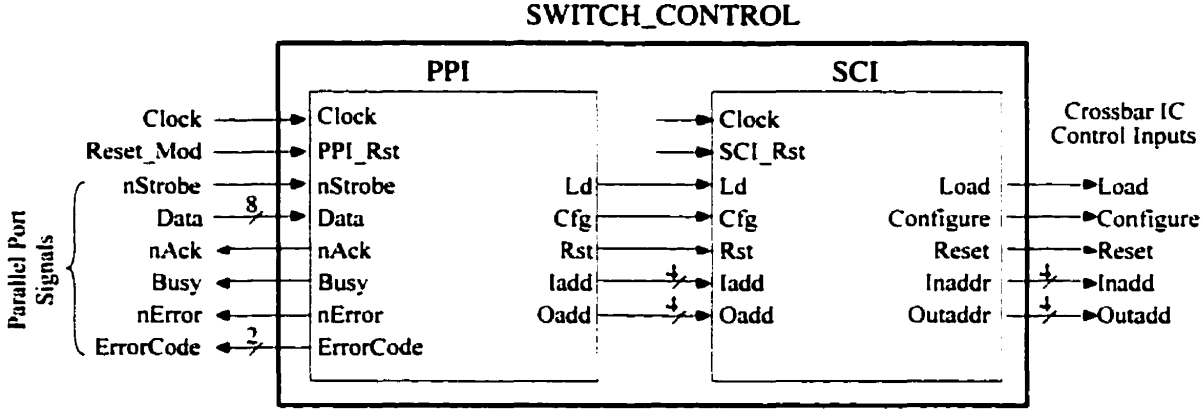


Fig. B.4. Switch Controller Block Diagram.

In Fig. B.4, signal DATA7..0 form the 8-bit datapath. Signal nSTROBE is driven by the Controller Workstation, and indicates there is valid data packet to be transferred to the Switch Core PCB. Signal BUSY is driven by the Switch Controller to indicate it is busy processing data. Signal nACK is driven by the Switch Controller to acknowledge it has received the data packet and indicates the end of the packet transfer.

Port Name	Port Type	Description
nACK	Output	Indicates to the Routing Software that the data packet is received (Active low)
BUSY	Output	Indicates to the Routing Software that the data transfer is being processed (Active high)
CLOCK	Input	Clock signal
CONFIGURE	Output	Crossbar IC's CONFIGURE input (Active high)
DATA[7..0]	Input	8-bit data packet transferred from Routing Software via the parallel port cable
nERROR	Output	Indicates to the Routing Software of an error in the data transfer (Active low)
ERRORCODE[1..0]	Output	2-bit error code (Refer to Table B.2)
INADD[3..0]	Output	Crossbar IC's INADD input
LOAD	Output	Crossbar IC's LOAD input (Active high)
OUTADD[3..0]	Output	Crossbar IC's OUTADD input
RESET	Output	Crossbar IC's RESET input (Active high)
RESET_MOD	Input	Initialize the Switch Controller (Active high)
nSTROBE	Input	Indicates to the Switch Controller of valid data on input DATA (Active low)

Table B.1. Summary of Switch Controller I/O Ports.

To fully configure the 16x16 optoelectronic switch core, the Switch Controller's PPI in Fig. B.4 receives 33 packets (32 address packets and 1 "configure command" packet) from the Routing Software. The 32 port addresses (4 bits each) are not stored in the Switch Controller, which would require at least 128 bits of memory. Therefore, each input-output port mapping received by the PPI is forwarded to the SCI for immediate loading into the appropriate crossbar IC Program Latches. The PPI responds independently to the address and command packets sent from the Routing Software. The input-output port mappings can be received in any sequence and an arbitrary number of times. But for each mapping transfer, the PPI imposes the sequence of an input port address packet followed by the output port address packet.

Referring to Fig. B.4, the Switch Controller uses the signal nERROR to report error conditions in the most recent data transfer. When nERROR=0, the Switch Controller returns a 2-bit code ERRORCODE1..0 to identify the error, as defined in Table B.2. An *Invalid OpCode Error* occurs when the OpCode field in the received packet does not represent an address type or switch command defined in Fig. B.3. A *Parity Error* occurs if the even parity check computed on the received packet fails. A *Sequence Error* occurs if the imposed transfer sequence of an input port address followed by an output port

address is violated.

ERRORCODE[1..0]	Description
0 0	Invalid OpCode
0 1	Parity Error
1 0	Sequence Error
1 1	No Error

Table B.2. Error Condition Definitions.

B.4. Parallel Port Interface Module

The Parallel Port Interface (PPI) module in Fig. B.4 handles the data transfer over the parallel port cable using the parallel port protocol described in Section B.6. The PPI is a 2-state finite state machine (FSM) and corresponds to the VHDL entity *ppi*, whose source code is found in Section B.7. The PPI module I/O are summarized in Table B.3. Output signals CFG, LD, RST, IADD and OADD are internally connected to the inputs of the SCI, as shown in the block diagram of Fig. B.4.

Port Name	Port Type	Description
nACK	Output	Acknowledges end of transaction (Active low)
BUSY	Output	PPI is busy processing data (Active high)
CFG	Output	Configures the crossbar IC (Active high)
CLOCK	Input	Clock signal
DATA[7..0]	Input	8-bit datapath of the parallel port cable
nERROR	Output	Error in data transfer (Active low)
ERRORCODE[1..0]	Output	2-bit error code (Refer to Table B.2)
IADD[3..0]	Output	4-bit input port address
LD	Output	Data available on IADD, OADD (Active high)
OADD[3..0]	Output	4-bit output port address
PPI_RST	Input	Initializes PPI's FSM (Active high)
RST	Output	Resets the crossbar IC (Active high)
nSTROBE	Input	Data available on DATA (Active low)

Table B.3. Summary of PPI Module I/O Ports.

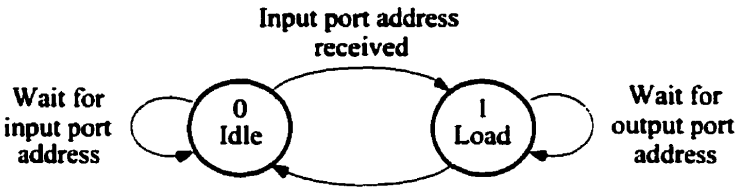


Fig. B.5. PPI Module Finite State Machine.

The PPI implements the FSM shown in Fig. B.5. The operations performed in **State 0** (Idle State) are described in the flowchart of Fig. B.6(a). The FSM goes into State 0 when the PPI is reset (PPI_RST=1). The PPI waits for data from the Routing Software by monitoring the debounced input nSTROBE. When nSTROBE=0 for 3 consecutive clock cycles, the PPI responds with BUSY=1, latches the 8-bit packet on input DATA7..0, and performs even parity check on the 8-bit packet using a 3-level XOR-tree. If there is no transmission error, the XOR computation result should be zero (0). Otherwise, the *Parity Error* is reported to the Routing Software. Then the PPI extracts the OpCode and Address fields from the 8-bit packet, and decodes the OpCode field according to the definitions in Fig. B.3 as follows:

- If the received packet contains an *input port address*, the 4-bit address is placed on the output datapath IADD to the SCI, and the FSM proceeds to State 1.
- Receiving an *output port address* generates a *Sequence Error*.
- Detecting a *Reset OpCode* sets RST=1 (for 1 clock cycle), which signals the SCI to clear the Program and Configuration latches in the crossbar ICs.
- Detecting a *Configure OpCode* sets CFG=1 (for 1 clock cycle), which signals the SCI to activate all the mappings in the switch core.
- Detecting an undefined OpCode generates an *Invalid OpCode Error*.

After processing, the FSM sets BUSY=0, sends nACK=0 pulse to end the transaction, and waits for more data from the Routing Software.

The operations performed in **State 1** (Load State) are described in Fig. B.6(b). The PPI waits for data from the Routing Software, by monitoring the debounced input nSTROBE. When nSTROBE=0 for 3 consecutive clock cycles, the PPI responds with BUSY=1, latches the 8-bit packet on input DATA, and performs even parity check on the packet. The OpCode field is extracted and decoded according to the definitions in Fig. B.3 as follows:

- Receiving an *input port address* or a *Configure OpCode* generates a *Sequence Error*.

- If the received packet contains an *output port address*, the 4-bit address is placed on output datapath OADD to the SCI, and the FSM asserts LD=1, which signals the SCI that both addresses are available on IADD and OADD.
- Detecting a *Reset OpCode* sets RST=1 (for 1 clock cycle).
- Detecting an undefined OpCode generates an *Invalid OpCode Error*.

After processing, the FSM sets BUSY=0, sends nACK=0 pulse and returns to State 0 to wait for other packets from the Routing Software.

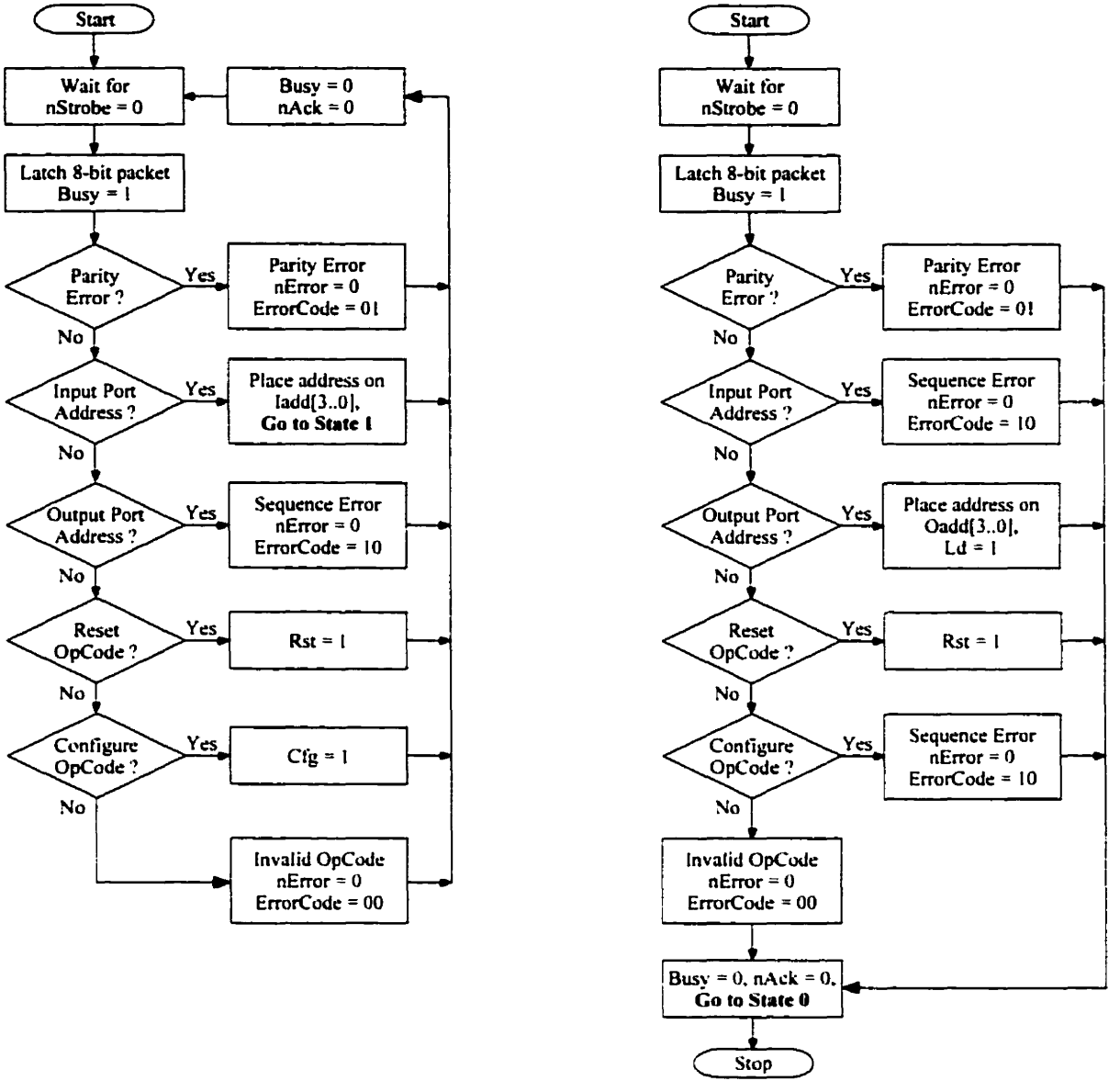


Fig. B.6. Flowcharts for PPI Module's FSM in (a) State 0, and (b) State 1.

B.5. Switch Configuration Interface Module

The Switch Configuration Interface (SCI) responds to instructions placed on its inputs by the PPI, and outputs the control signals to perform the hardware configuration of the 8 crossbar switch ICs.

Since each crossbar switch IC on the Switch Core PCB operates on a 1-bit slice of the byte-wide datapath from the 16 input ports, all the crossbar switch ICs are configured identically and simultaneously with the same routing mappings.

The SCI is a 4-state finite state machine (FSM) and corresponds to the VHDL entity *sci*, whose source code is found in Section B.7. The SCI module I/O are summarized in Table B.4.

Port Name	Port Type	Description
CFG	Input	Configures crossbar IC (Active high)
CLOCK	Input	Clock signal
CONFIGURE	Output	Crossbar IC's CONFIGURE input (Active high)
IADD[3..0]	Input	4-bit input port address
INADDR[3..0]	Output	Crossbar IC's INADD input
LD	Input	Data available on IADD and OADD (Active high)
LOAD	Output	Crossbar IC's LOAD input (Active high)
OADD[3..0]	Input	4-bit output port address
OUTADDR[3..0]	Output	Crossbar IC's OUTADD input
RESET	Output	Crossbar IC's RESET input (Active high)
RST	Input	Resets crossbar IC (Active high)
SCI_RST	Input	Initializes SCI's FSM (Active high)

Table B.4. Summary of SCI Module I/O Ports.

Input IADD3..0 receives the 4-bit input port address from the PPI. Input OADD3..0 receives the 4-bit output port address from the PPI. Input LD requests the SCI to configure the mapping specified by IADD and OADD into the switch core. Input CFG activates the downloaded switch configuration for the entire switch core. Input RST instructs the SCI to clear the 128 Program Latches and 128 Configuration Latches in the 8 crossbar switch ICs which make up the electrical switch core. As a result, input port 0 of the switch core is broadcasted to all the 16 output ports. The 5 outputs of the SCI drive the corresponding control and address inputs of the 8 crossbar switch ICs, as described in Table B.4.

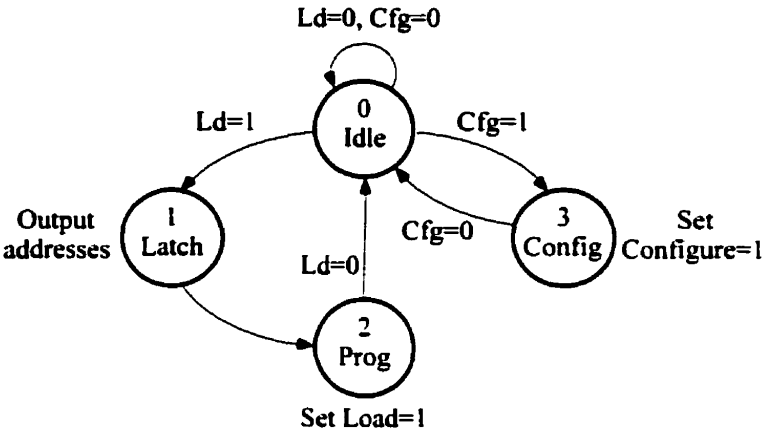


Fig. B.7. SCI Module Finite State Machine.

The SCI implements the FSM shown in Fig. B.7 and is described below. State transitions occur on the rising edge of the clock signal.

- The FSM goes into **State 0** (Idle State) when the SCI is reset (SCI_RST=1). In State 0, if input RST=1, the FSM outputs RESET=1, which drives the crossbar switch IC RESET input. The FSM remains in State 0 until LD=1 or CFG=1.
- In **State 1** (Latch State), the SCI latches the port addresses on inputs IADD and OADD, forwards them to the crossbar switch ICs, and proceeds to State 2.
- In **State 2** (Prog State), the SCI outputs LOAD=1, which drives the crossbar switch IC LOAD input, and returns to State 0 when input LD=0. The 4-bit input port address is loaded into the Program Latches selected by the 4-bit output port address.
- In **State 3** (Config State), the SCI outputs CONFIGURE=1, which drives the crossbar switch IC CONFIGURE input, and returns to State 0 when input CFG=0. The Configuration Latches in the crossbar switch ICs load the content of the Program Latches, hence activating the new switch configuration.

B.6. Parallel Port Protocol

The communication protocol used to transfer data between the controller workstation and the Switch Controller is based on the Parallel Port Protocol [60]. The original parallel port protocol was developed to control the printer from the host computer, and supports a bi-directional 8-bit datapath and 9 control signals. Our communication protocol differs slightly from the original protocol and implements an uni-directional 8-bit datapath with 6 control signals. The subsequent discussion presents the hardware and software programming components of the parallel port protocol, including the physical port signal definitions, the parallel port I/O registers on the workstation, the data transfer algorithm and the protocol timing.

B.6.1. Signal Definitions

The parallel port signals defined in our implementation are summarized in Table B.5. The Routing Software controls signal nSTROBE and the datapath DATA[7..0], while the Switch Controller drives the remaining 5 control signals. The column "Port Invert" means the signal is inverted by the parallel port circuitry, and it will be of opposite polarity when accessed by the Routing Software. The pin numbers in Table B.5 correspond to the physical pin positions on a 25-pin parallel port ribbon cable connector of Fig. B.8.

Pin No.	Signal Name	Description	Port Invert
1	nSTROBE	Data available (Active low)	Yes
2 - 9	DATA[0..7]	8-bit data lines	
10	nACK	End of transfer (Active low)	
11	BUSY	Busy signal (Active high)	Yes
12	ERRORCODE[1]	Error code MSB (Refer to Table 3.2)	
13	ERRORCODE[0]	Error code LSB (Refer to Table B.2)	
15	nERROR	Transfer error (Active low)	
18 - 25	Ground (0V)		

Table B.5. Parallel Port Signal Definitions.

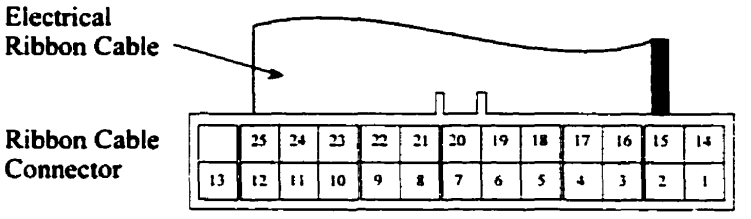


Fig. B.8. Pin Assignment on Parallel Port Ribbon Cable Connector.

B.6.2. I/O Addresses and Registers

On the PC workstation, the primary parallel port LPT1 is assigned the fixed I/O address 378h (in hexadecimal notation). Relative to this base address, there are 3 byte-wide I/O registers: Data Register (address 378h), Status Register (address 379h) and Control Register (address 37Ah). Each parallel port signal is represented by a bit in one of the I/O registers, as shown in Table B.6.

Address	Register Name	Bit No.	Signal	Pin No.
378h	Data Register (Read/Write)	7	DATA[7]	9
		6	DATA[6]	8
		5	DATA[5]	7
		4	DATA[4]	6
		3	DATA[3]	5
		2	DATA[2]	4
		1	DATA[1]	3
		0	DATA[0]	2
379h	Status Register (Read only)	7	BUSY (inv.)	11
		6	nACK	10
		5	ERRORCODE[1]	12
		4	ERRORCODE[0]	13
		3	nERROR	15
		2 - 0	Unused	
37Ah	Control Register (Write only)	7 - 1	Unused	
		0	nSTROBE (inv.)	1

Table B.6. Register Bit Assignments (bit 7 is MSB of register).

The Routing Software can access the parallel port signals by reading or writing to the I/O registers. The *Data Register* is used by the Routing Software for writing (and

reading) 8-bit data lines DATA[7..0] at pins 2-9. The read-only *Status Register* is used by the Switch Controller on the Switch Core PCB to report the communication and error status of the data transfer. The Routing Software can initiate a data transfer using bit 0 of the write-only *Control Register*.

Recall from Table B.5 that BUSY and nSTROBE are hardware inverted by the parallel port circuitry. This means that when the signal BUSY (pin 11) is driven high (+5V) by the Switch Controller, bit 7 of the Status Register reads logic 0 and when it is driven low (0V), bit 7 reads logic 1. For the signal nSTROBE, when a logic 1 is written into bit 0 of the Control Register, pin 1 is driven low (0V), and when a logic 0 is written into bit 0, pin 1 is driven high (+5V).

B.6.3. Operating Principles and Timing

The parallel port protocol is implemented under software control. The Routing Software sends 33 packets (32 address packets and 1 "command" packet) to the Switch Controller to configure the 16x16 optoelectronic switch core. The timing diagram for the transfer of an address packet is shown in Fig. B.9. The event diagram of the protocol is shown in Fig. B.10.

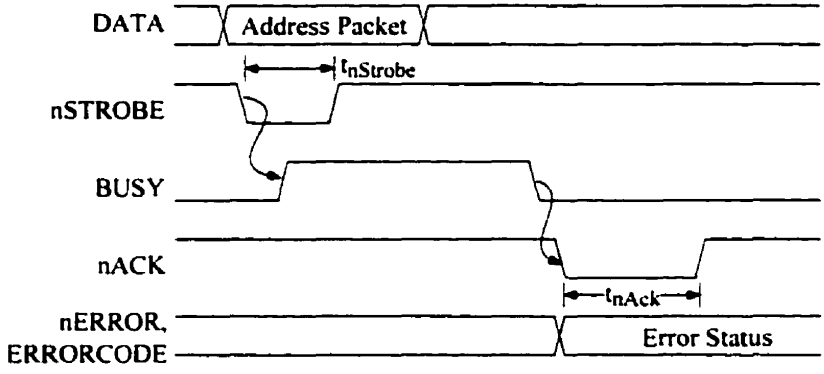


Fig. B.9. Parallel Port Protocol Timing Diagram.

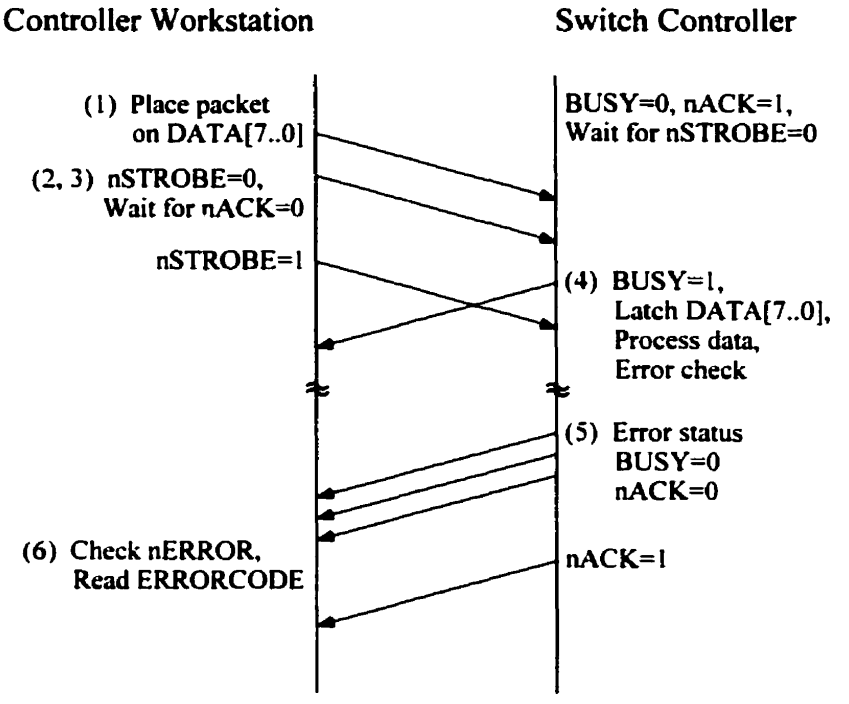


Fig. B.10. Parallel Port Protocol Event Diagram.

Referring to Fig. B.10, the Routing Software on the Controller Workstation initiates the data transfer and the protocol proceeds as described in the following steps:

1. The Routing Software begins the transaction by writing the 8-bit packet into the Data Register and onto DATA[7..0].
2. Then, the Routing Software checks the BUSY signal by reading Status Register bit 7 and waits until BUSY=0 (i.e., bit 7 is "1"), meaning the Switch Controller is ready to receive data.
3. The Routing Software sets nSTROBE=0, by writing logic 1 into Control Register bit 0, for duration $t_{nStrobe} > 1$ µs to inform the Switch Controller that data is available on the data lines. The Routing Software monitors the nACK signal by reading Status Register bit 6.
4. After detecting nSTROBE=0, the Switch Controller latches DATA[7..0] and responds with BUSY=1, which sets Status Register bit 7 to logic 0.
5. Once the data has been received and processed, the Switch Controller sets BUSY=0 and acknowledges the end of the transaction with a nACK=0 pulse of duration $t_{nAck} > 5$ µs. Normally, the packet is received without error. The Switch Controller reports the error status using nERROR and ERRORCODE.
6. When the Routing Software receives nACK=0, it checks the error status by

reading Status Register bits 3-5.

7. If no there is no error (nERROR=1), another transaction can begin from step 1.
8. If nERROR=0, the Routing Software reports the error identified in ERRORCODE to the user through the graphical user interface. Further transactions are halted and the data transfer must be restarted.

B.7. VHDL Source Code

In this section, the VHDL source code for the following entities are provided:

- entity *switch_control* implements the top level Switch Controller.
- entity *ppi* implements the Parallel Port Interface within the Switch Controller.
- entity *sci* implements the Switch Configuration Interface within the Switch Controller.

B.7.1. Switch Controller

```
--------------------------------------------------------------------------
-- Filename    :       switch_control.vhd
-- Entity      :       switch_control
-- Title       :       FPGA-based Crossbar Switch Controller
--
-- Author      :       Albert AU
-- Date        :       Feb-24-2000
-- Revision    :       3.0
--
-- Description:        This is the top-level entity file for the
--                     SWITCH CONTROLLER on the Switch Core PCB.
--                     The Switch Controller contains the Parallel
--                     Port Interface (PPI) and the Switch Config-
--                     uration Interface (SCI). The Switch Controller
--                     handles the data transfer between the Controller
--                     Workstation and the Switch Core PCB, and
--                     implements the control logic for configuring
--                     the crossbar switch ICs, which make up the
--                     switch core.
--
-- References  :       The Switch Controller is described in detail
--                     in Sections 4.2.2 and B.3. The block diagram of
--                     the Switch Controller is shown in Fig. B.4.
--                     Synthesis results are described in Section 4.2.3.
--                     The VHDL code is adapted from the work described
--                     in reference [30].
--------------------------------------------------------------------------

-- Use IEEE Standard Logic libaries
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

-- The Switch Controller I/O are defined
-- Detail description follows the declaration
ENTITY switch_control IS
PORT (
        Clock           : IN std_logic;
        Reset_Mod       : IN std_logic;

-- I/O specific to the parallel port
        nStrobe         : IN std_logic;
        Data            : IN std_logic_vector( 7 DOWNTO 0 );
        nAck            : OUT std_logic;
        Busy            : OUT std_logic;
        nError          : OUT std_logic;
        ErrorCode       : OUT std_logic_vector( 1 DOWNTO 0 );
```

```
-- I/O specific to switch configuration
       Load           : OUT std_logic;
       Configure      : OUT std_logic;
       Reset          : OUT std_logic;
       Inaddr         : OUT std_logic_vector( 3 DOWNTO 0 );
       Outaddr        : OUT std_logic_vector( 3 DOWNTO 0 )
) ;
END switch_control ;

-------------------------------------------------------------
-- DESCRIPTION OF INPUT-OUTPUT PORTS (See Appendix B)
--
-- The first column titled "Pin" is the FPGA device pin number.
-- An asterix symbol * next to the pin number denotes the FPGA
-- pin is hardwired on the PCB. The second column titled
-- "Signal" is the VHDL I/O declared above. The third column
-- gives the description.
-------------------------------------------------------------
-- Pin Signal         Description
-------------------------------------------------------------
-- F3* Clock          clock signal (from oscillator on PCB)
-- F18 Reset_Mod      initialize the Switch Controller

-- Parallel Port I/O
-- W5  nStrobe        valid data on data lines
-- V5  Data0          data bit 0
-- U5  Data1          data bit 1
-- R2* Data2          data bit 2
-- P2* Data3          data bit 3
-- N2* Data4          data bit 4
-- M2* Data5          data bit 5
-- L2* Data6          data bit 6
-- K2* Data7          data bit 7
-- J2* nAck           packet received
-- H2* Busy           processing transfer
-- W6  nError         error flag
-- G2* ErrorCode1     error code bit 1
-- F2* ErrorCode0     error code bit 0

-- Switch Configuration I/O
-- C5  Load           connects to crossbar's LOAD input
-- C4  Configure      connects to crossbar's CONFIGURE input
-- C3  Reset          connects to crossbar's RESET input
-- B6  Inaddr0        input port address bit 0
-- A6  Inaddr1        input port address bit 1
-- B5  Inaddr2        input port address bit 2
-- A5  Inaddr3        input port address bit 3
-- B9  Outaddr0       output port address bit 0
-- A9  Outaddr1       output port address bit 1
-- B8  Outaddr2       output port address bit 2
-- A8  Outaddr3       output port address bit 3
-------------------------------------------------------------

-- Stitch the PPI and SCI modules structurally. The PPI and SCI
-- modules are described in Section B.7.
ARCHITECTURE structure OF switch_control IS

--------------------------
-- Parallel Port Interface
--------------------------
COMPONENT ppi
PORT (
       Clock          : IN std_logic;
```

```
        PPI_Rst         : IN std_logic;

        nStrobe         : IN std_logic;
        Data            : IN std_logic_vector( 7 DOWNTO 0 );
        nAck            : OUT std_logic;
        Busy            : OUT std_logic;
        nError          : OUT std_logic;
        ErrorCode       : OUT std_logic_vector( 1 DOWNTO 0 );

        Ld              : OUT std_logic;
        Cfg             : OUT std_logic;
        Rst             : OUT std_logic;
        Iadd            : OUT std_logic_vector( 3 DOWNTO 0 );
        Oadd            : OUT std_logic_vector( 3 DOWNTO 0 )
) ;
END COMPONENT ;

---------------------------------
-- Switch Configuration Interface
---------------------------------
COMPONENT sci
PORT (
        Clock           : IN std_logic;
        SCI_rst         : IN std_logic;

        Ld              : IN std_logic;
        Cfg             : IN std_logic;
        Rst             : IN std_logic;
        Iadd            : IN std_logic_vector( 3 DOWNTO 0 );
        Oadd            : IN std_logic_vector( 3 DOWNTO 0 );

        Load            : out std_logic;
        Configure       : out std_logic;
        Reset           : out std_logic;
        Inaddr          : out std_logic_vector( 3 DOWNTO 0 );
        Outaddr         : out std_logic_vector( 3 DOWNTO 0 )
) ;
END COMPONENT ;

-- Define some internal signals to
-- interconnect PPI and SCI modules.

SIGNAL  ld_sig, cfg_sig, rst_sig : std_logic;
SIGNAL iadd_sig, oadd_sig : std_logic_vector( 3 DOWNTO 0 );

-------------------------------
-- Define internal organization
-------------------------------

BEGIN
        -- Define connections of PPI ports to
        -- top-level I/O ports or internal signals.

        PPI_Module: ppi PORT MAP (
                Clock           => Clock,
                PPI_Rst         => Reset_Mod,

                nStrobe         => nStrobe,
                Data            => Data,
                nAck            => nAck,
                nError          => nError,
                Busy            => Busy,
                ErrorCode       => ErrorCode,
```

```
                Ld              => ld_sig,
                Cfg             => cfg_sig,
                Rst             => rst_sig,
                Iadd            => iadd_sig,
                Oadd            => oadd_sig
        ) ;

        -- Define connections of SCI ports to
        -- top-level I/O ports or internal signals.

        SCI_Module: sci PORT MAP (
                Clock           => Clock,
                SCI_rst         => Reset_Mod,

                Ld              => ld_sig,
                Cfg             => cfg_sig,
                Rst             => rst_sig,
                Iadd            => iadd_sig,
                Oadd            => oadd_sig,

                Load            => Load,
                Configure       => Configure,
                Reset           => Reset,
                Inaddr          => Inaddr,
                Outaddr         => Outaddr
        ) ;

END structure ;
```

B.7.2. Parallel Port Interface

```
--------------------------------------------------------------------------
-- Filename    :      ppi.vhd
-- Entity      :      ppi
-- Title       :      Parallel Port Interface (PPI)
--
-- Author      :      Albert Au
-- Date        :      Mar 18, 2000
-- Revision    :      2.1
--
-- Description:       The Parallel Port Interface (PPI) is a submodule
--                    within the Switch Controller. The PPI interface
--                    to the Routing Software on the Controller
--                    Workstation. The PPI is a finite state machine
--                    (FSM) which implements the parallel port protocol.
--                    The PPI supplies data to the Switch Configuration
--                    Interface (SCI).
--
-- References  :      The PPI is described in detail in Section B.4.
--                    The parallel port protocol is described in
--                    Section B.6. The FSM state diagram is shown in
--                    Fig. B.5. Synthesis results are presented in
--                    Section 4.2.3. The VHDL code is adapted from the
--                    work described in reference [30].
--------------------------------------------------------------------------

-- Use IEEE Standard Logic libraries.
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

-- The Parallel Port Interface I/O are defined here.
-- Detail description follows the declaration.
ENTITY ppi IS
PORT (
       Clock          : IN std_logic;
       PPI_Rst        : IN std_logic;

-- I/O specific to the parallel port protocol
       nStrobe        : IN std_logic;
       Data           : IN std_logic_vector( 7 DOWNTO 0 );
       nAck           : OUT std_logic;
       Busy           : OUT std_logic;
       nError         : OUT std_logic;
       ErrorCode      : OUT std_logic_vector( 1 DOWNTO 0 );

-- Control signals to SCI
       Ld             : OUT std_logic;
       Cfg            : OUT std_logic;
       Rst            : OUT std_logic;
       Iadd           : OUT std_logic_vector( 3 DOWNTO 0 );
       Oadd           : OUT std_logic_vector( 3 DOWNTO 0 )
) ;
END ppi ;

-------------------------------------------------------------
-- DESCRIPTION OF INPUT-OUTPUT PORTS (See Appendix B)
--
-- The first column titled "Signal" is the VHDL I/O declared
-- above. The second column gives the description.
-------------------------------------------------------------
--     Signal         Description
```

```
------------------------------------------------------------

--      Clock           clock signal
--      PPI_Rst         initialize PPI state machine

-- Parallel Port signals
--      nStrobe         valid data on data lines
--      Data            8-bit data
--      nAck            packet received
--      nError          error flag
--      ErrorCode       error code

-- Control signals to SCI
--      Ld              Load port mapping
--      Cfg             Activate switch configurations
--      Rst             Reset switch core
--      Iadd            4-bit input port address
--      Oadd            4-bit output port address
------------------------------------------------------------


ARCHITECTURE fsm OF ppi IS

-------------------------------
-- Define internal organization
-------------------------------

-- Opcode Definitions (Section B.2.2)
-- Data is transferred between the Controller Workstation and
-- the Switch Controller 8 bits at a time. The data format is
-- shown in Fig. B.3, and contains an even parity bit, 3-bit
-- opcode field and 4-bit address field.

-- Opcode 001 identifies the address field as an input port.
-- Opcode 010 identifies the address field as an output port.
-- Opcode 011 is the command to configure the entire switch core.
-- Opcode 100 is the command to reset the switch core.

Constant INPUT_ADDR  : STD_LOGIC_VECTOR( 2 DOWNTO 0 ) := "001";
Constant OUTPUT_ADDR : STD_LOGIC_VECTOR( 2 DOWNTO 0 ) := "010";
Constant CONFIG_COMM : STD_LOGIC_VECTOR( 2 DOWNTO 0 ) := "011";
Constant RESET_COMM  : STD_LOGIC_VECTOR( 2 DOWNTO 0 ) := "100";


-- Error Condition Definitions (Section B.3)
-- When there is an error in the most recent data transfer, the
-- PPI reports the error condition using the 2-bit signal ErrorCode.

-- Error code 00 is an Invalid Opcode Error: the Opcode in the
-- received packet is invalid.
-- Error code 01 is a Parity Error: the even parity computed
-- on the received packet failed.
-- Error code 10 is a Sequence Error: an input port address packet
-- must be followed by an output port address packet.
-- Error code 11 means No Error.

Constant INVALID_OPCODE : STD_LOGIC_VECTOR( 1 DOWNTO 0 ) := "00";
Constant PARITY_ERROR   : STD_LOGIC_VECTOR( 1 DOWNTO 0 ) := "01";
Constant SEQ_ERROR      : STD_LOGIC_VECTOR( 1 DOWNTO 0 ) := "10";
Constant NO_ERROR       : STD_LOGIC_VECTOR( 1 DOWNTO 0 ) := "11";


BEGIN
```

```
PROCESS( nStrobe, PPI_Rst, Clock )

-- There are 2 states in the FSM, as shown in the state diagram of Fig. B.5.
--      state 0 is Idle      Wait for an input port address, reset command
--                           or configure command from the Routing Software.
--                           If input port address is received, proceed
--                           to Load state. If reset command is received,
--                           request the SCI to reset the crossbars. If
--                           configure command is received, request the SCI
--                           to configure the crossbars.
--                           Return to this state if the FSM is reset.
--      state 1 is Load      Wait for output port address or reset command
--                           from the Routing Software. If output port address
--                           is received, request the SCI to load the mapping
--                           into the crossbars and return to Idle state.
--                           If reset command is received, request the SCI to
--                           reset the crossbars.

TYPE state_type IS (Idle, Load);
VARIABLE state : state_type;

-- Define some internal signals.
VARIABLE temp_reg : std_logic_vector( 7 downto 0 ) ;
ALIAS OpCode : std_logic_vector(2 downto 0) IS temp_reg( 6 downto 4 );
ALIAS Addr   : std_logic_vector(3 downto 0) IS temp_reg( 3 downto 0 );
VARIABLE even_parity_check : std_logic ;

VARIABLE nStrobe_debounced : STD_LOGIC;
VARIABLE nStrobe_DFF : STD_LOGIC_VECTOR( 0 TO 2 );
VARIABLE pre_Busy : STD_LOGIC;

BEGIN

-- Reset (asynchronous) the FSM ?
IF( PPI_Rst = '1' ) THEN
       -- Initialize the FSM; proceed to Idle state.
       -- Ensure debounced version of input nStrobe is inactive.
       state := Idle ;
       nStrobe_DFF := (OTHERS=>'1');

ELSIF rising_edge(Clock)  THEN

       -- Debounce active low input nStrobe using 3 DFFs in series.
       nStrobe_DFF(1 TO 2) := nStrobe_DFF(0 TO 1);
       nStrobe_DFF(0) := nStrobe;
       nStrobe_debounced :=  nStrobe_DFF(0) OR nStrobe_DFF(1) OR
                             nStrobe_DFF(2);

-------------------------------------
-- Define state 0 of FSM
-- Refer to flowchart in Fig. B.6(a)
-------------------------------------

       CASE state IS
       WHEN Idle =>
              IF (nStrobe_debounced = '0' AND pre_Busy = '0') THEN
              -- There is new data from the Routing Software and
              -- the PPI is currently not doing anything. So we
              -- should process the incoming packet ...

                     temp_reg := Data;      -- Latch 8-bit packet from the data lines
                     nAck <= '0';           -- Acknowledge receipt of new packet
                     pre_Busy := '1';       -- Working on new packet
```

```
-- Verify EVEN parity of the received packet using
-- 3-level XOR tree.
even_parity_check :=
       (( temp_reg(7) XOR temp_reg(6) ) XOR
        ( temp_reg(5) XOR temp_reg(4) )) XOR
       (( temp_reg(3) XOR temp_reg(2) ) XOR
        ( temp_reg(1) XOR temp_reg(0) ));

IF ( even_parity_check = '0' ) THEN
       -- No parity error, so now check the
       -- Opcode field in the packet.

       IF ( OpCode = INPUT_ADDR ) THEN
              -- Input address received.
              -- Send input port address to the SCI and
              -- proceed to Load state.
              Iadd <= Addr;
              state := Load; -- State change

       ELSIF ( OpCode = OUTPUT_ADDR ) THEN
              -- Output address received.
              -- Report Sequence Error and remain in Idle state.
              Iadd <= (OTHERS=>'0');
              Oadd <= (OTHERS=>'0');
              state := Idle;
              nError <= '0'; -- Error flag
              ErrorCode <= SEQ_ERROR; -- Sequence Error

       ELSIF ( OpCode = CONFIG_COMM ) THEN
              -- Configure Command received.
              -- Request SCI to activate the mappings for the
              -- entire switch core and remain in Idle state.
              Iadd <= (OTHERS=>'0');
              Oadd <= (OTHERS=>'0');
              Cfg <= '1'; -- Configure request
              state := Idle;

       ELSIF ( OpCode = RESET_COMM ) THEN
              -- Reset Command received.
              -- Request SCI to initialize all crossbar ICs
              -- and remain in Idle state.
              Iadd <= (OTHERS=>'0');
              Oadd <= (OTHERS=>'0');
              Rst <= '1'; -- Reset request
              state := Idle;

       ELSE
              -- Invalid opcode detected.
              -- Report Invalid OpCode Error and remain in
              -- Idle state.
              Iadd <= (OTHERS=>'0');
              Oadd <= (OTHERS=>'0');
              nError <= '0'; -- Error flag
              ErrorCode <= INVALID_OPCODE; -- Invalid Opcode
              state := Idle;

       END IF; -- end of checking opcodes

ELSE
       -- Parity Error detected in the received packet.
       -- Report Parity Error and remain in Idle state.
       Iadd <= (OTHERS=>'0');
       Oadd <= (OTHERS=>'0');
       nError <= '0'; -- Error flag
```

```
                        ErrorCode <= PARITY_ERROR; -- Parity Error
                        state := Idle;

                END IF ; -- end of parity

        ELSE
        -- There is no new data from the Routing Software,
        -- so we continue to wait in the Idle state ...

                IF (nStrobe_debounced = '1') THEN
                        nAck <= '1';
                        pre_Busy := '0';
                        nError <= '1';
                        ErrorCode <= NO_ERROR; -- No Error
                END IF;

                Ld <= '0';
                Cfg <= '0';
                Rst <= '0';
                state := Idle;
        END IF ; -- end of nStrobe is 0

-------------------------------------
-- Define state 1 of FSM
-- Refer to flowchart in Fig. B.6(b)
-------------------------------------

        WHEN Load  =>
                IF (nStrobe_debounced = '0' AND pre_Busy = '0') THEN
                -- There is new data from the Routing Software and
                -- the PPI is currently not doing anything. So we
                -- should process the incoming packet ...

                        temp_reg := Data;       -- Latch 8-bit packet from the data lines
                        nAck <= '0';            -- Acknowledge receipt of new packet
                        pre_Busy := '1';        -- Working on new packet

                        -- Verify EVEN parity of the received packet using
                        -- 3-leve XOR tree.
                        even_parity_check :=
                                (( temp_reg(7) XOR temp_reg(6) ) XOR
                                 ( temp_reg(5) XOR temp_reg(4) )) XOR
                                (( temp_reg(3) XOR temp_reg(2) ) XOR
                                 ( temp_reg(1) XOR temp_reg(0) ));

                        IF ( even_parity_check = '0' ) THEN
                                -- No parity error, so now check the
                                -- Opcode field in the packet.

                                IF ( OpCode = INPUT_ADDR ) OR
                                   ( OpCode = CONFIG_COMM ) THEN
                                        -- Input address or Configure command received.
                                        -- Report Sequence Error and return to Idle.
                                        Iadd <= (OTHERS=>'0');
                                        Oadd <= (OTHERS=>'0');
                                        nError <= '0'; -- Error flag
                                        ErrorCode <= SEQ_ERROR; -- Sequence Error
                                        state := Idle;

                                ELSIF ( OpCode = OUTPUT_ADDR ) THEN
                                        -- Output address received.
                                        -- Send output port address to the SCI. Both
                                        -- addresses are available, so request SCI to
                                        -- load the mapping into the crossbar ICs.
```

```
                                        -- Then return to Idle state.
                                        Oadd <= Addr;
                                        Ld <= '1'; -- Load request
                                        state := Idle;

                                ELSIF ( OpCode = RESET_COMM ) THEN
                                        -- Reset Command received.
                                        -- Request SCI to initialize all crossbar ICs
                                        -- and return to dle state.
                                Iadd <= (OTHERS=>'0');
                                Oadd <= (OTHERS=>'0');
                                Rst <= '1'; -- Reset request
                                state := Idle;

                                ELSE
                                        -- Invalid opcode detected.
                                        -- Report Invalid OpCode Error and return to
                                        -- Idle state.
                                        Iadd <= (OTHERS=>'0');
                                        Oadd <= (OTHERS=>'0');
                                        nError <= '0'; -- Error flag
                                        ErrorCode <= INVALID_OPCODE; -- Invalid Opcode
                                        state := Idle;

                                END IF; -- end of checking opcodes

                        ELSE
                                -- Parity Error detected in the received packet.
                                -- Report Parity Error and return to Idle state.
                                Iadd <= (OTHERS=>'0');
                                Oadd <= (OTHERS=>'0');
                                nError <= '0'; -- Error flag
                                ErrorCode <= PARITY_ERROR; -- Parity Error
                                state := Idle;

                        END IF; -- end of parity

                ELSE
                -- There is no new data from the Routing Software,
                -- so we continue to wait in the Load state ...

                        IF (nStrobe_debounced = '1') THEN
                                nAck <= '1';
                                pre_Busy := '0';
                                nError <= '1';
                                ErrorCode <= NO_ERROR;
                        END IF;

                        Ld <= '0';
                        Cfg <= '0';
                        Rst <= '0';
                        state := Load;
                END IF; -- end of nStrobe is 0

        END CASE; -- end of state machine

END IF; -- end of rising_edge(Clock)

Busy <= pre_Busy;

END PROCESS;

END fsm;
```

B.7.3. Switch Configuration Interface

```
--------------------------------------------------------------------
-- Filename    :      sci.vhd
-- Entity      :      sci
-- Title       :      Switch Configuration Interface (SCI)
--
-- Author      :      Albert Au
-- Date        :      Mar-17-2000
-- Revision    :      2.2
--
-- Description:       The Switch Configuration Interface (SCI) is a
--                    submodule within the Switch Controller. The SCI
--                    receives routing data from the PPI and configures
--                    the Triquint TQ8017 crossbar switch ICs, which
--                    make up the switch core on the Switch Core PCB.
--                    The SCI is a finite state machine (FSM), which
--                    drives the address and control inputs of the
--                    crossbar switch ICs.
--
-- References  :      The SCI is described in detail in Section B.5.
--                    The FSM state diagram is shown in Fig. B.7.
--                    Synthesis results are presented in Section 4.2.3.
--                    The VHDL code is adapted from the work described
--                    in reference [30].
--------------------------------------------------------------------

-- Use IEEE Standard Logic libraries
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

-- The Switch Configuration Interface I/O are defined here.
-- Detail description follows the declaration.
ENTITY sci IS
PORT (
       Clock          : IN std_logic;
       SCI_rst        : IN std_logic;

-- Control signals from PPI
       Ld             : IN std_logic;
       Cfg            : IN std_logic;
       Rst            : IN std_logic;
       Iadd           : IN std_logic_vector( 3 DOWNTO 0 );
       Oadd           : IN std_logic_vector( 3 DOWNTO 0 );

-- These I/O drive the crossbar IC input pins
       Load           : OUT std_logic;
       Configure      : OUT std_logic;
       Reset          : OUT std_logic;
       Inaddr         : OUT std_logic_vector( 3 DOWNTO 0 );
       Outaddr        : OUT std_logic_vector( 3 DOWNTO 0 )
);
END sci;

------------------------------------------------------------
-- DESCRIPTION OF INPUT-OUTPUT PORTS (See Appendix B)
--
-- The first column titled "Signal" is the VHDL I/O declared
-- above. The second column gives the description.
------------------------------------------------------------
--     Signal         Description
------------------------------------------------------------
```

```
--      Clock           clock signal
--      SCI_Rst         initialize SCI state machine

-- Control signals from PPI
--      Ld              Load port mapping
--      Cfg             Activate switch configurations
--      Rst             Reset switch core
--      Iadd            input port address
--      Oadd            output port address

-- I/O to drive crossbar IC input pins
--      Load            connect to crossbar's LOAD
--      Configure       connect to crossbar's CONFIGURE
--      Reset           connect to crossbar's RESET
--      Inaddr          4-bit input port address
--      Outaddr         4-bit output port address
-------------------------------------------------------------


ARCHITECTURE fsm OF sci IS

-------------------------------
-- Define internal organization
-------------------------------

BEGIN

PROCESS( clock, ld )

-- There are 4 states in the FSM, as shown in the
-- state diagram of Fig. B.7.
--      state 0 is Idle         Wait for PPI instructions. Return to
--                              this state if the FSM is reset.
--      state 1 is Latch        Latch port mapping (input and output
--                              addresses) supplied by the PPI.
--      state 2 is Prog         Load the designated Program Latch in
--                              the crossbar ICs with the designated
--                              input port address.
--      state 3 is Config       Load all the Configuration Latches
--                              to activate all port mappings.

TYPE state_type IS (Idle, Latch, Prog, Config);
VARIABLE state : state_type;

-- Define some internal signals.
VARIABLE myinaddr  : std_logic_vector( 3 DOWNTO 0 );
VARIABLE myoutaddr : std_logic_vector( 3 DOWNTO 0 );

BEGIN
        -- Reset (asynchronous) the FSM ?
        IF( SCI_rst = '1') THEN
                -- Initialize the FSM; proceed to Idle state.
                state := Idle;

        ELSIF( rising_edge(Clock) ) THEN

                -- Reset the switch core ?
                IF( Rst = '1' ) THEN
                        -- Initialize all crossbar switch ICs by asserting
                        -- their RESET input.
                        Reset <= '1';
                ELSE
                        Reset <= '0';
```

```
        END IF;

-----------------------------------------
-- Define the FSM operations and outputs
-----------------------------------------

        CASE state IS
        WHEN Idle =>
            IF (Ld = '1') THEN
            -- There is a request to load a port mapping
            -- into the switch core. Grab the input port
            -- and output port addresses from the PPI.
            -- Proceed to Latch state.
                myinaddr := Iadd;
                myoutaddr := Oadd;
                state := Latch;

            ELSIF (Cfg = '1') THEN
            -- There is a request to configure the entire
            -- switch core, which is handled in Config state.
                state := Config;
            END IF;

            Inaddr      <= myinaddr;
            Outaddr     <= myoutaddr;
            Load        <= '0';
            Configure   <= '0';

        WHEN Latch =>
            -- To load a port mapping into the switch core,
            -- we first drive the input port and output port
            -- addresses to the crossbar switch ICs. Then
            -- proceed to Prog state.
            state := Prog;

            Inaddr      <= myinaddr;   -- Input address to crossbar
            Outaddr     <= myoutaddr;  -- Output address to crossbar
            Load        <= '0';
            Configure   <= '0';

        WHEN Prog =>
            -- After driving out the input port and output
            -- port addresses, we then assert (to 1) the Load
            -- input of the crossbar switch ICs.
            -- Return to the Idle state when the request is
            -- complete (signal Ld).

            IF (Ld = '0') THEN
                state := Idle;
            END IF;

            Inaddr      <= myinaddr;   -- Input address to crossbar
            Outaddr     <= myoutaddr;  -- Output address to crossbar
            Load        <= '1';        -- Assert crossbar pin Load
            Configure   <= '0';

        WHEN Config =>
            -- To configure the entire switch core, thus
            -- activating the port mappings, we assert (to 1)
            -- the Configure input of the crossbar switch ICs.
            -- Return to the Idle state when the request is
            -- complete (signal Cfg).

            IF (Cfg = '0') THEN
```

```
                        state := Idle;
                    END IF;

                    Inaddr          <= myinaddr;
                    Outaddr         <= myoutaddr;
                    Load            <= '0';
                    Configure       <= '1';        -- Assert crossbar pin Configure
                END CASE;

        END IF; -- end of rising_edge(clock)

END PROCESS;

END fsm;
```

APPENDIX C

Daughterboard Pinout Schematics

This appendix provides the daughterboard pinout schematics for the following IC devices, which are used in the NIC and switch core prototypes:

- PECL signal buffer IC
- Optobus transceiver IC
- Transmitter and receiver ICs
- Crossbar switch IC

The pinout schematics map the *IC device pins* on the IC package to the *daughterboard pins* mounted under the daughterboard. The following explains the conventions used in the pinout schematics of Figs. C.1 to C.4:

- Device I/O pin positions are shown as shaded circles or rectangles.
- Daughterboard pins are drawn as white circles.
- The numbers or names displayed within the daughterboard pins represent the device I/O pin numbers or signal names.
- Unused daughterboard pins are left blank.
- C*n* denotes the location of a discrete capacitor, where *n* is an integer. The positive polarity is indicated by the + sign.
- R*n* denotes the location of a discrete resistor, where *n* is an integer. The positive polarity is indicated by the + sign.
- SIP-R*n* denotes the location of 50 Ω SIP terminating resistor pack, where *n* is an integer. Section 3.1.2 describes the SIP terminating resistor pack. SIP terminating resistor packs have connections to power (5V) and ground (0V), and can contain 8 or 10 pins. The numbers or names displayed in the terminating resistor packs are the device I/O signals being terminated.

C.1. PECL Signal Buffer IC

The Motorola 10E416 PECL signal buffer IC is used within the OTM on the NIC and switch core. The OTM is described in Section 3.2.4. The electrical specifications of the PECL signal buffer IC are provided in [53].

The top view (i.e., component side) of the PECL signal buffer IC daughterboard is shown in Fig. C.1. The PECL signal buffer IC is placed into a PLCC-to-PGA socket, which is attached to the daughterboard. In Fig. C.1, the device I/O signal names are displayed on the daughterboard pins. Signals D0..D4 are the 5 differential inputs and Q0..Q4 are the 5 differential outputs. VBB is the switching reference voltage (3.7 V) used for AC-coupled or single-ended inputs. VCC4 is the 5V power supply connection to internal logic. VCC1, VCC2 and VCC3 are 5V power supply connections to output drivers. VEE is the most negative supply voltage level at 0V. These power-ground pins are bypassed using capacitors C1 to C4, which are explained below:

- Power supply VCC1 is bypassed to VEE using 0.1 µF capacitor C1.
- Power supply VCC2 is bypassed to VEE using 0.1 µF capacitor C2.
- Power supply VCC3 is bypassed to VEE using 0.1 µF capacitor C3.
- Power supply VCC4 is bypassed to VEE using 0.1 µF capacitor C4.

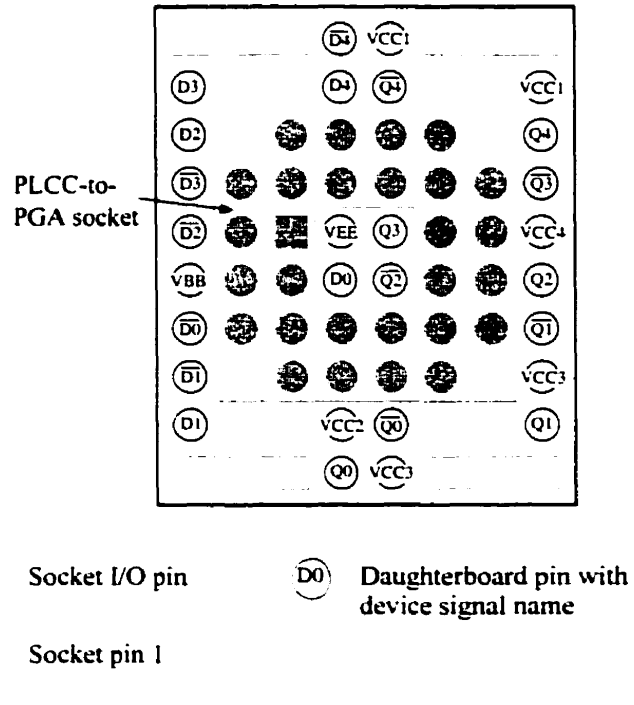


Fig. C.1. PECL Signal Buffer IC Daughterboard Pinout (Top View).

C.2. Optobus Transceiver IC

The Motorola Optobus IC is used within the OTM on the NIC and switch core. The OTM is described in Section 3.2.4. The electrical specifications of the Optobus IC are provided in [65].

The top view (i.e., component side) of the Optobus IC daughterboard is shown in Fig. C.2. The Optobus transmit subsystem is located on the left and the receive subsystem is located on the right.

In Fig. C.2, the device I/O signal names are displayed on the daughterboard pins. Signals D0..D9 are the 10 differential PECL inputs, and are terminated by SIP-R1, SIP-R2 and SIP-R3. Signals Q0..Q9 are the 10 differential CML outputs, which require external 50 Ω pull-up resistors to 5V.

Power (5V) and ground (0V) pins are specified as follows: GT*n* denotes transmitter ground, VT*n* denotes transmitter power, GR*n* denotes receiver ground, and VR*n* denotes receiver power, where *n* is the row number of the power-ground pin as defined in [65]. Table C.1 explains the bypassing scheme for the Optobus power-ground pins:

Bypass Capacitors	Capacitances	Power-ground Pins
C1, C2	0.1 µF, 5 µF	VR2, GR1
C3, C4	0.1 µF, 5 µF	VR4, GR3
C5, C6	0.1 µF, 5 µF	VR6/7, GR5
C7, C8	0.1 µF, 5 µF	VR9/10, GR8
C9, C10	0.1 µF, 5 µF	VR13/14, GR12
C11, C12	0.1 µF, 5 µF	VT3, GT1
C13, C14	0.1 µF, 5 µF	VT6, GT4
C15, C16	0.1 µF, 5 µF	VT10, GT8
C17, C18	0.1 µF, 5 µF	VT12, GT13/14

Table C.1. Optobus IC Bypass Capacitors.

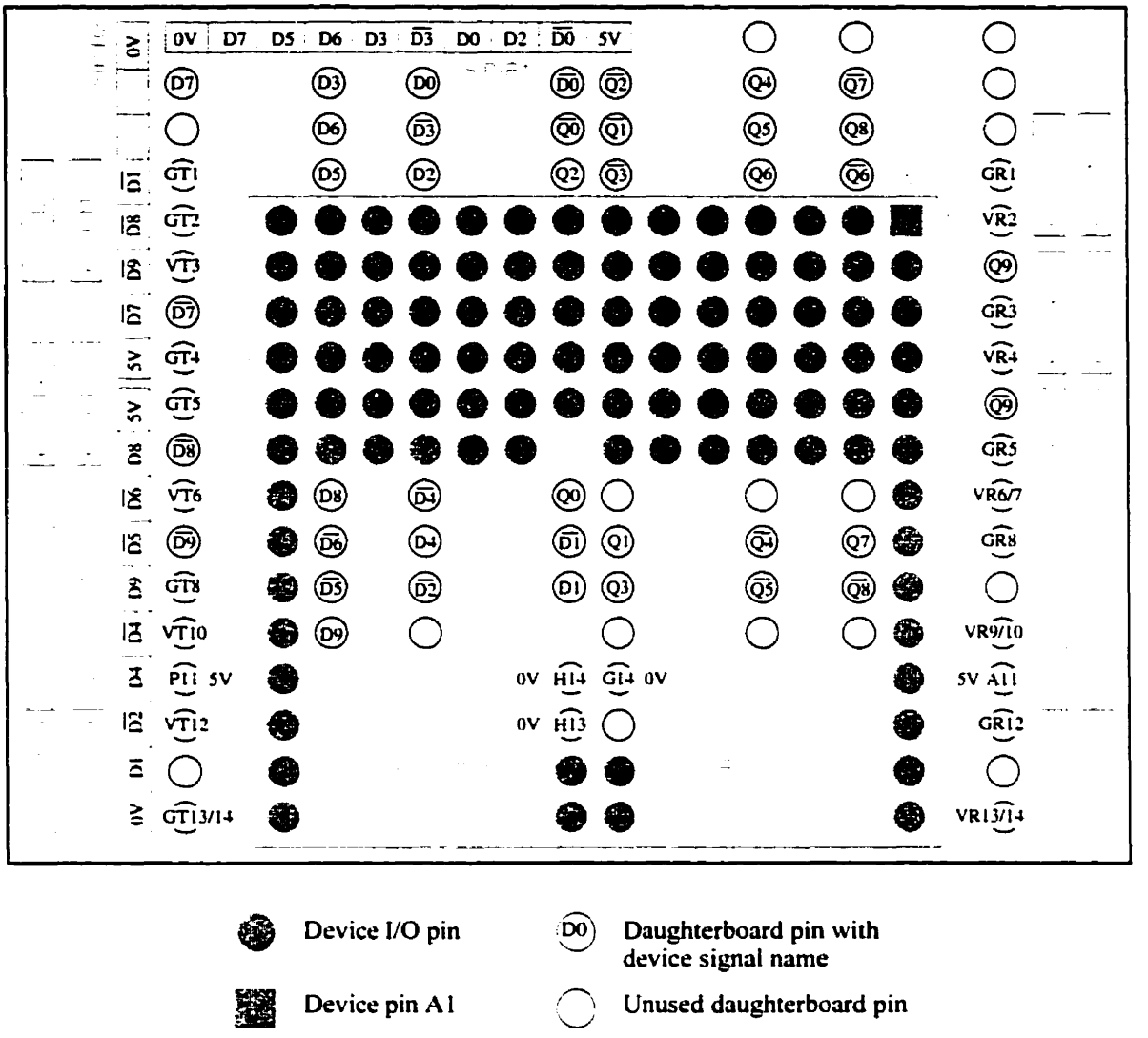


Fig. C.2. Motorola Optobus IC Daughterboard Pinout (Top View).

C.3. Transmitter and Receiver IC

The Hewlett-Packard HDMP-1012 gigabit transmitter IC implements the NIC's Transmitter Module, which is described in Section 3.2.2. The HDMP-1014 gigabit receiver IC implements the NIC's Receiver Module, which is described in Section 3.2.3. The electrical specifications of the transmitter and receiver ICs are provided in [49].

The top view (i.e., component side) of the transmitter and receiver ICs daughterboard is shown in Fig. C.3, where the transmitter IC occupies the right half and the receiver IC occupies the left side. The device I/O pin numbers are displayed on the daughterboard pins. The mappings between the device I/O pin numbers and the I/O signal names are provided in [49]. The following explains the bypassing and terminating schemes for the transmitter IC in Fig. C.3:

- Single-ended PECL inputs are terminated by SIP-R3 through SIP-R7.
- Capacitor C9 is the 0.1 μF PLL integrator capacitor described in Section 3.2.2.2, and is wired to pins 1 & 2 and 3 & 4.
- Power supply pins 23 & 24 are bypassed to ground pins 21 & 22 using the 0.1 μF capacitor C4 in series with the 10 Ω damping resistor R4.
- Power supply pins 43 & 44 are bypassed to ground pins 41 & 42 using the 0.1 μF capacitor C5 in series with the 10 Ω damping resistor R5.
- Power supply pins 63 & 64 are bypassed to ground pins 61 & 62 using the 0.1 μF capacitor C6 in series with the 10 Ω damping resistor R6.

The following explains the bypassing and terminating schemes for the receiver IC in Fig. C.3:

- Single-ended PECL inputs are terminated by SIP-R1 and SIP-R2.
- Capacitor C8 is the 0.1 μF PLL integrator capacitor described in Section 3.2.3.2, and is wired to pins 1 & 2 and 3 & 4.
- Power supply pins 23 & 24 are bypassed to ground pins 21 & 22 using the 0.1 μF capacitor C1 in series with the 10 Ω damping resistor R1.
- Power supply pins 43 & 44 are bypassed to ground pins 41 & 42 using the 0.1 μF capacitor C2 in series with the 10 Ω damping resistor R2.
- Power supply pins 63 & 64 are bypassed to ground pins 61 & 62 using the 0.1 μF capacitor C3 in series with the 10 Ω damping resistor R3.
- Pin 32 is pulled to 5V through the 10 Ω damping resistor R7.
- Pin 72 is pulled to 5V through the 10 Ω damping resistor R8.

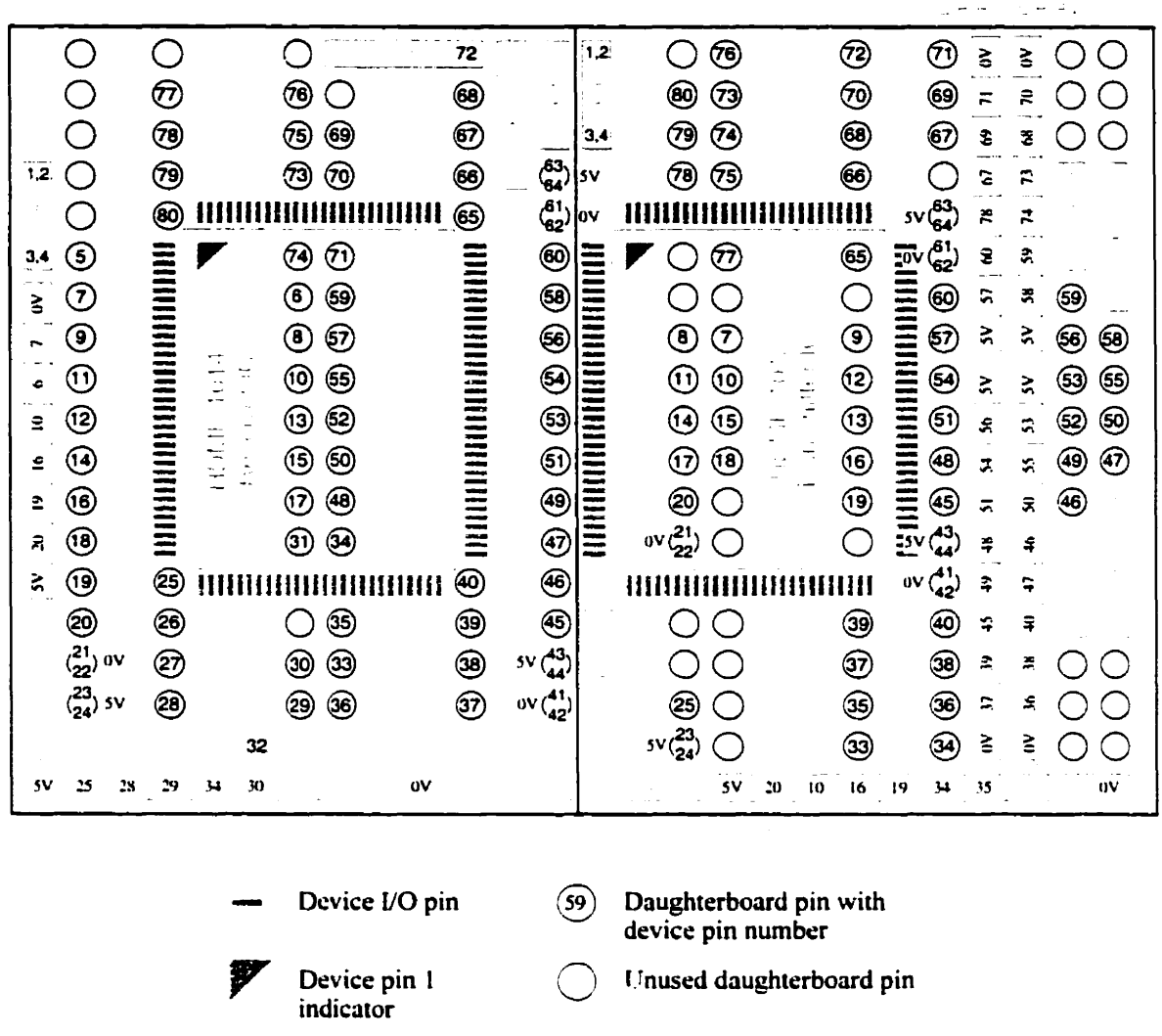


Fig. C.3. Transmitter and Receiver ICs Daughterboard Pinout (Top View).

C.4. Crossbar Switch IC

The Triquint TQ8017 crossbar switch IC is used on the prototype switch core, and is described in Section 4.1.1. The electrical specifications of the crossbar switch IC are provided in [36].

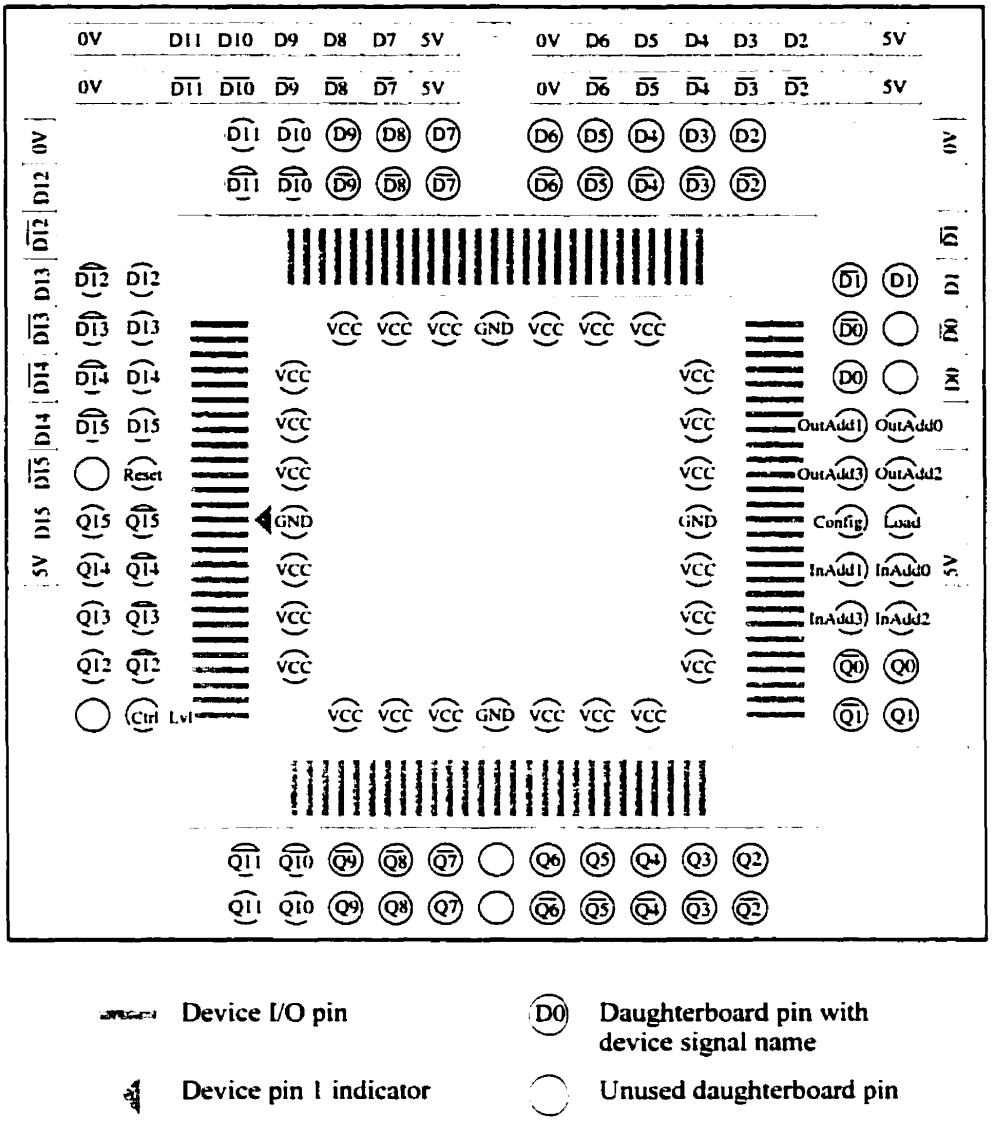


Fig. C.4. Crossbar Switch IC Daughterboard Pinout (Top View).

The top view (i.e., component side) of the crossbar switch IC daughterboard is shown in Fig. C.4. The device I/O signal names are displayed on the daughterboard pins. Signals D0..D15 are the 16 differential PECL input ports, and are terminated by SIP-R1

to SIP-R6. Signals Q0..Q15 are the 16 differential PECL output ports. The power pins (VCC, 5V) connect to a power plane on the daughterboard's top layer and the ground pins (GND, 0V) connect to a ground plane on the daughterboard's bottom layer. These planes are bypassed using the 0.1 μF capacitors C1 to C5 shown in Fig. C.4.