CHAPTER 4.

AN INTERFACE TO HIGH-SPEED MULTIPLE-ACCESS CHANNELS

4.1. General

Presented in this chapter the design of a serial electronic interface to a multiple-access fiber-optic network. Its goal is to maximize the interface bandwidth (greater than 1 gigabit/sec with GaAs technology), which is usually the bottleneck of an optical communication system ([OfFa87a]).

The design has five major objectives: (1) To maximize the transmission rate by reducing the critical timing path to one flipflop and one gate. (2) To minimize the number of flipflops which are directly driven by the transmission clock, so as to decrease clock skew. (3) To decode and perform the serial-to-parallel conversion of very small packets without the training period associated with a phase-locked loop. (4) To minimize the ratio between the transmission clock frequency and the baud rate.

These objectives are achieved by two techniques: (1) Hierarchical interface design – the interface design has four levels which successively reduce the data path width from the host bus to the optical link. (2) By using the 8B/12B conservative encoding scheme, as described in Chapter 3, characterized by a fixed number of transitions in every codeword. These transitions are used directly for decoding and for serial-to-parallel conversion, without having to recover the clock by means of a phased-locked loop.

Three general assumptions guide this design: (1) Hardware is not a scarce resource; it can be used to reduce the complexity of the system's operation and to improve its performance. For example, data encoding and CRC generation can be done on blocks of bits, not serially. (2) The medium bandwidth is higher than the digital electronic interface bandwidth. The parallel–to–serial conversion (a shift register with parallel load), and the decoder with serial–to–parallel conversion are the bottlenecks of the digital electronic interface. (3) The lines which connect the discrete devices are transmission lines with very accurate characteristic impedance, and these can be used as highly accurate and stable *delay lines*. The delays for this design, which are less than one nanosecond, can be realized by less than 20 cm of transmission line on a printed circuit board.

4.2. Comparison of the Conservative Code with Other Codes

The interface is designed for using the conservative code. In this section this code is compared with other codes. The major criterion is maximizing the bandwidth of the digital electronic interface, which is more critical than the bandwidth of the optical channel. Therefore, the objective is to have the DBBRR (digital bandwidth to baud rate ratio) as low as possible, i.e., to have maximum baud rate in a given technology characterized by typical and worst–case gate delay. Three criteria are used to evaluate different codes: (1) Resolution – the required sensitivity of the receiver for correct decoding, expressed in terms of a bit–cell. The resolution is the reciprocal value of the DBBRR. (2) DC level – the difference between the time that the transmitted signal is high and the time that the signal is low; it is desirable that this difference be as close as possible to zero. (3) Decoding without a PLL. One of the design objectives is to be able to decode an incoming packet without the training period required by a PLL, which can be substantial at high transmission rates.

The following is a review of some of the relevant self–clocking codes [Seve80], [Sore84]:

(1) Manchester Encoding – is a level type code in which a one has a high level at the beginning of the bit–cell with high–to–low transition at midcell, while a zero starts at a low level with low–to–high transition at midcell. Note that this code is not invertible.

(2) Biphase–Mark Encoding – is an edge–type code. Each bit–cell begins with an edge; for a one an additional edge occurs at midcell, while for a zero there is no additional edge. The code is invertible, since there is no rule for the polarities of the edges.

(3) Miller Encoding – is also called "delay modulation." This is an edge–type code; each one in the serial data is encoded as a mid–cell transition, while zero either has no transition (following a one) or is encoded with an edge at the beginning of the bit–cell. As a result, the encoded serial data have edges occurring at intervals of 1.0, 1.5 or 2.0 bit times.

(4) The 4B/5B encoding schemes – This code changes 4 bits of data into 5 bits of data. The motivation for this encoding scheme is to ensure enough transitions but not a fixed number. It is possible to choose a subset of 24 words such that no more than three consecutive zeros occur. If the bits are transmitted as non–return to zero inverted (NRZI), then there is no signal transition separation of more than three–bit cell periods. This scheme is almost balanced [JoIy84] and

has an efficiency of 80%, or bandwidth increase of only 25%.

The following table summarizes some relevant properties of the different encoding schemes:

TABLE 4.1. CODING SCHEMES COMPARISON

Scheme	DC	Efficiency	Decoding without PLL
Manchester/Biphase	none	50%	possible
Miller	almost none	50%	difficult
1–2 PWM (average)	almost none	67%	very simple
1–2 PWM (worst)	almost none	50%	very simple
NRZ	substantial	100%	impossible
4B/5B (with NRZI)	almost none	80%	very difficult
Conservative (8B/12B)	almost none	67%	very simple
Conservative (16B/20B)	almost none	80%	simple

The 8B/12B conservative code has three basic advantages: (i) the efficiency is better than 67%, (ii) the dc level is very low with a very high probability, and (iii) the decoding is very simple without a PLL.

4.3. Bandwidth Matching

In order to match the serial link and host bandwidths, the interface is hierarchically designed with four stages for managing the data path from the host bus to the optical link. At each stage the data path is controlled by a finite state machine. The requirement for matching is that between adjacent stages the stage closer to the host will be fast enough to serve the stage closer to the optical link.

Figure 4.1 describes the interface stages in terms of two major characteristics: (i) the technology required for realizing the stage, and (ii) the maximum time period (or interval) allowed for the transition from state to state. The following list specifies the main operation at each stage:

(1) Shift register level – single–bit interval, the time for changing the state of the shift register from shift to parallel load.

(2) Encoding/decoding – 8–bit interval, for performing these functions in parallel on data bytes.

(3) Transmitter and receiver finite state controllers – 64–bit interval, the time for writing or reading a word to/from the transmitter or the receiver.

(4) Buffer management, routing and access control unit – packet length interval for performing these functions in real time.

4.4. Functional Description

This section outlines the basic properties of the interface, as shown in Figure 4.2, and describes the principles of its operation. It is assumed that there is a set of buffers (or FIFOs) which can each store one packet of data. The buffers are dual–ported to the host bus and to the interface (receiver or transmitter). Access control, routing control, and buffer management are not included in the following description.

4.4.1. Principle of operation

The communication system has three major parts, as shown in Figure 4.3. The central part is the communication channel, which carries serial data. The channel transfers the serial bit stream from the encoder to the decoder.

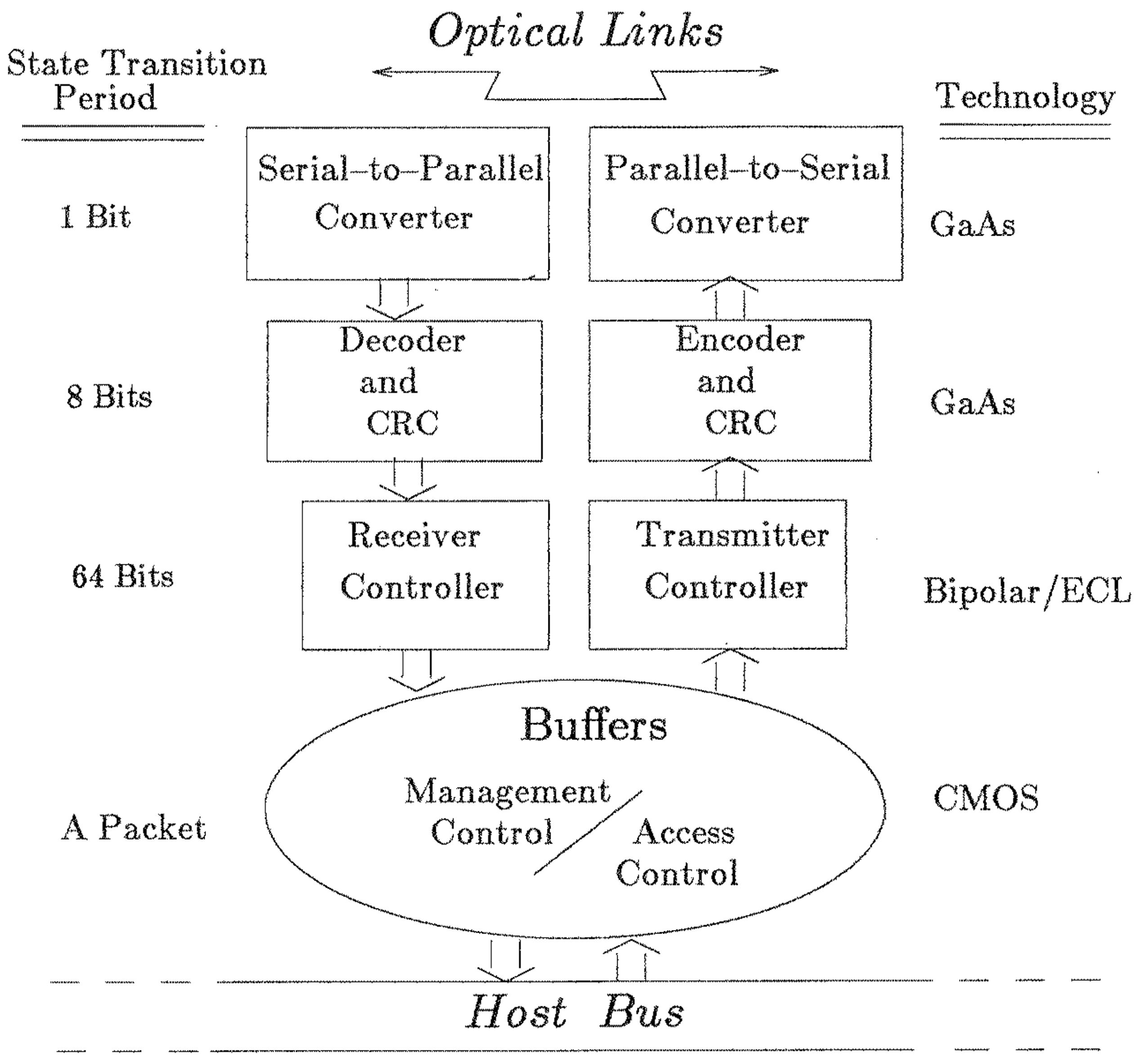

Figure 4.1: Bandwidth Matching

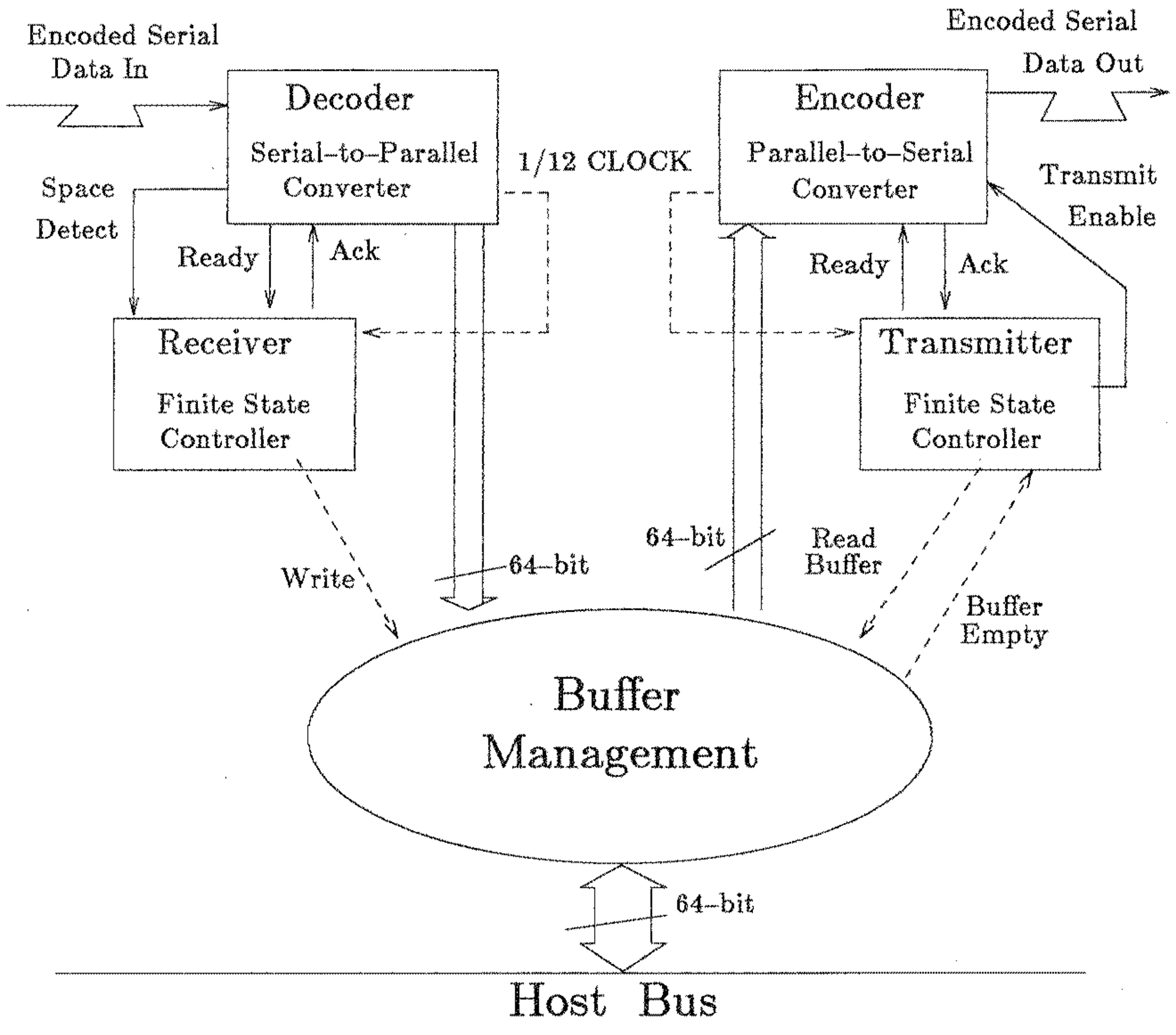

Figure 4.2: Functional Description of the Interface

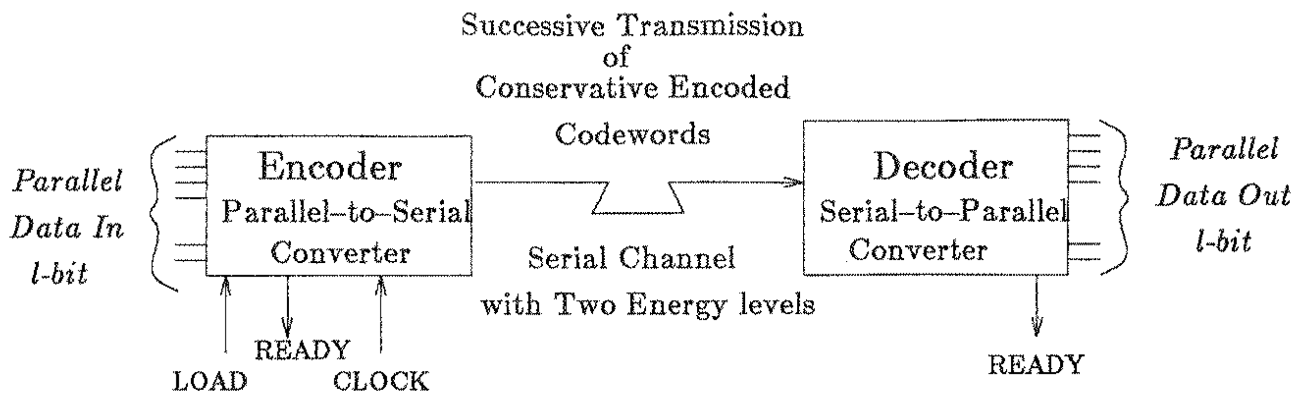
Successive Transmission of Conservative Encoded Codewords
Parallel Data In l-bit
Encoder
Parallel–to–Serial Converter
LOAD
READY
CLOCK
Serial Channel with Two Energy levels
Decoder
Serial–to–Parallel Converter
READY
Parallel Data Out l-bit

Figure 4.3: Communication Block Diagram

Note that the decoder and serial–to–parallel converter operate without explicitly recovering the receiving clock.

4.4.1.1. Clocking

A basic question is whether to use one or both edges of the clock signal in order to get better system performance. In general, clock generator modules have the following properties: (1) a clock stability and accuracy better than $\pm 0.1\%$, and (2) a clock symmetry accuracy of $\pm 5\%$. In high speed serial communication, in order to improve the reliability of the decoding and parallel–to–serial conversion, it is important that the output signal be as accurate as possible. In this design *only one of the two clock edges is used,* so that the output timing will not depend on clock symmetry. Therefore, the digital interface bandwidth is greater than or equal to the baud rate.

4.4.1.2. The packet header

The data to be transferred are organized into packets, which are continuous serial streams of bits. The packet header consists of one word of n bits; the first three quarters of these have some pattern of zeros and ones, and the last quarter of the header is a space, as shown in Figure 4.4. The receiver searches for this space and starts recording data only after it is detected. This technique reduces the effect of random noise, which might occur while the optical receiver is idle.

4.4.1.3. Full–duplex interface

The transmitter and receiver are implemented as two independent submodules, i.e., the network interface is *full-duplex*. This makes the design of the interface simple, reliable, and testable. An important consequence of this, for topologies like star, is that a receiver may check its transmitter output, using either an error–detecting code or direct packet–to–packet comparison. A full–duplex interface is necessary for collision detection in random access protocols, such as CSMA/CD.

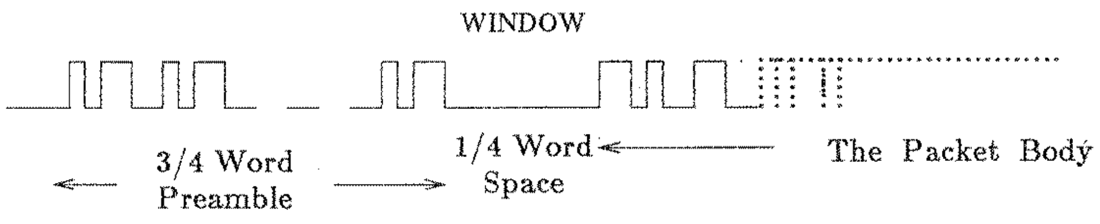

Figure 4.4: The Packet Head Pattern

4.4.2. The transmitter

The functional description of the transmitter is shown in Figure 4.5. The input register is reloaded, and the next word is encoded while the shift register transmits the current word.

The transmitter's data path is controlled by two finite state machines which perform the following functions: (1) Generate the packet header. (2) Encode the data in parallel on blocks of 8 bits by a combinational network. (3) Parallel–to–serial conversion. Load Next Word and Ready are the handshake signals between the transmitter finite state controller and the encoder.

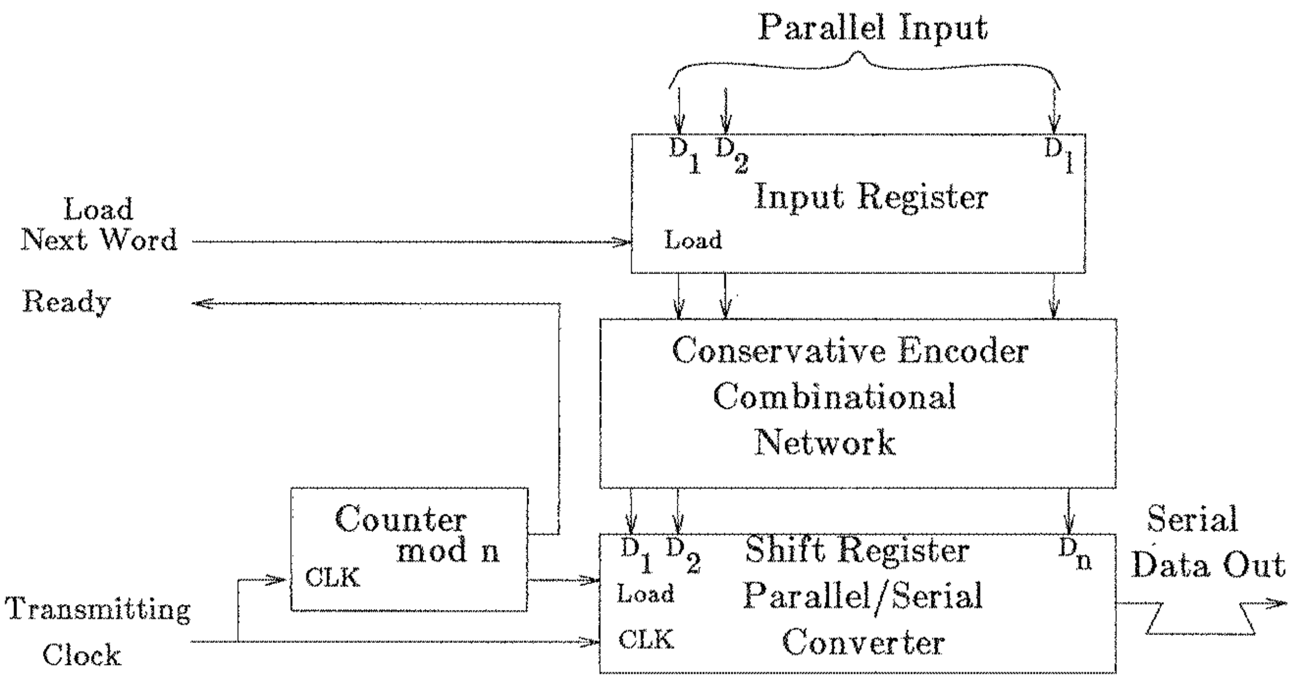

Figure 4.5: Functional Description of the Transmitter

4.4.3. The receiver

The functional description of the receiver is as shown in Figure 4.6. The receiver's data path is likewise controlled by two finite state machines, which perform the following functions: (1) Decode and serial–to–parallel conversion. (2) Check the correctness of the incoming packet. (3) Packet destination recognition; if the packet is not addressed to this node, the receiver discards it.

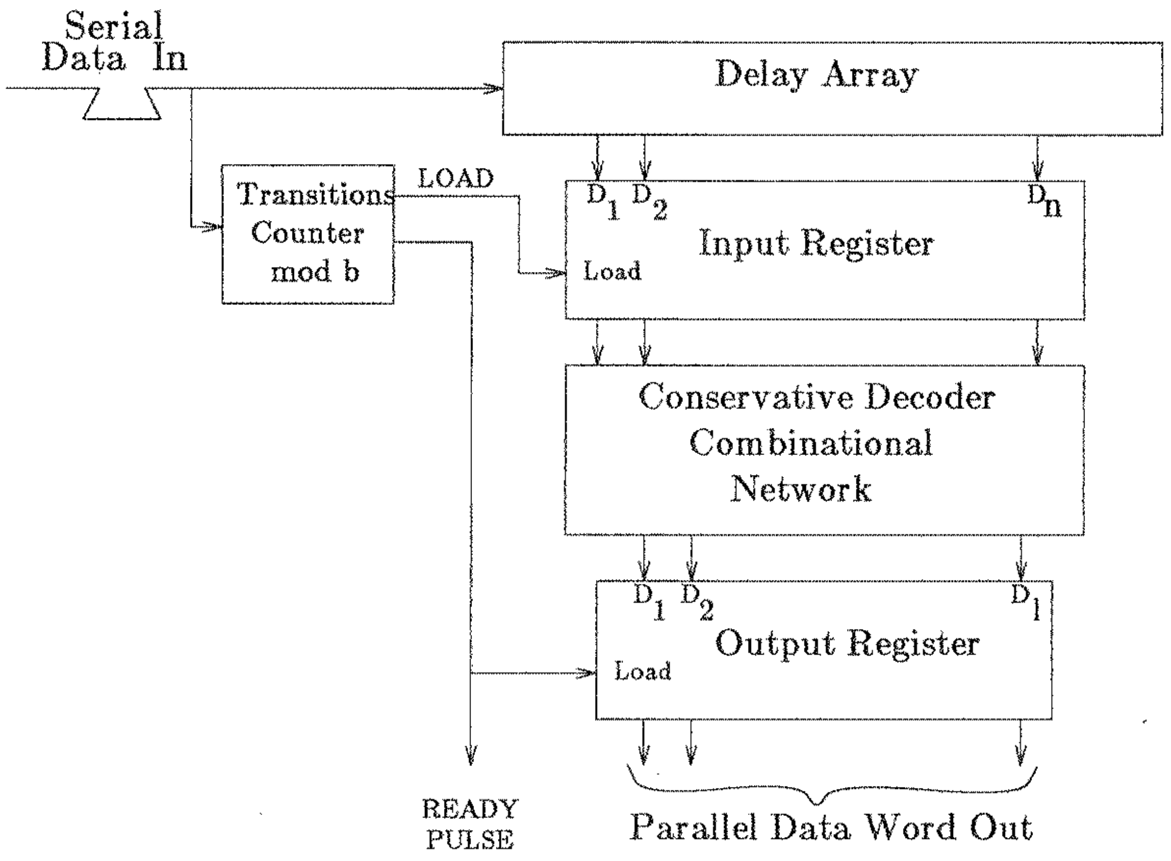

Figure 4.6: Functional Description of the Receiver

4.5. Encoder and Parallel–to–Serial Converter

Described in this section the details of the link level interface for either the 1–2 PWM or 8B/12B conservative encoding schemes. It could be implemented with GaAs technology and use a 1500 MHz clock for a transmission rate of 1 gigabit/sec.

In order to obtain maximum speed, the critical delay path is minimized to a shift register with a parallel load (see Figure 4.7), i.e., the *critical delay* is a linear summation (worst case analysis) of three delays: flipflop set–up time, t_s, flipflop propagation delay, t_d, and NOR–gate propagation delay, t_n. The first two delays are necessary for the shift register, and the third delay is required for the parallel load. The maximum frequency at which the interface can operate is therefore

$$BW_{\max}=\frac{1}{t_s+t_d+t_n}$$

Note that loading via a single gate with a wired–OR output is possible because the shift register contains only zeros by the time of the next load.

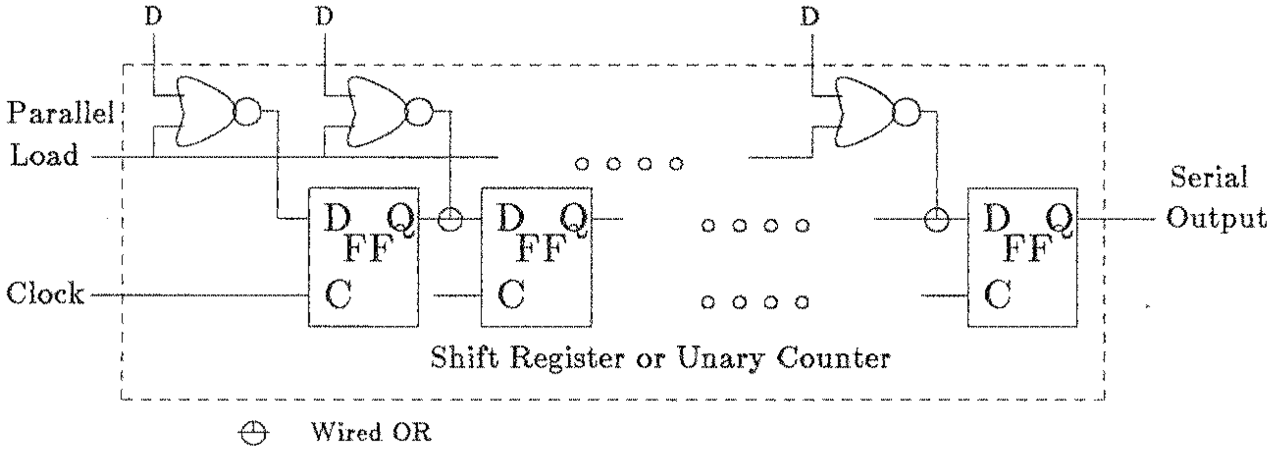

Figure 4.7: The Shift Register Design

Figure 4.8 (p. 57) is a block diagram of the transmitter's link interface. The **Shift Register Array** is part of the bandwidth matching mechanism. It is parallel–loaded with a 64–bit word from the **Data Bus**. The 8 least significant (rightmost) bits of the **Shift Register Array** are input to two **Combinational Networks**. The **Shift Register Array** is clocked by the /LOAD–BYTE signal. The **Word Load Unary Counter** counts eight /LOAD–BYTE pulses, on the 8th of which LOAD–WORD signal is asserted. As a result, on the following /LOAD–BYTE pulse, the array is loaded with a new word from the **Data Bus**.

One of the **Combinational Networks** is the encoder, which receives 8 bits from the shift register array, and transforms them into a 12–bit codeword. If the START–ENABLE signal is deasserted, the output of this network is always zero. The other combinational network is the **Programmable Load Timer**, which receives the same 8 bits from the array and transforms them into an output word with a single 1, whose position determines the time of the next byte load. The output of this combinational network is loaded into the **Byte Load Unary Counter**.

In order to achieve maximum performance all counters are **unary**, i.e., shift registers with parallel–load (as shown in Figure 4.7). There are three unary counters in this interface: (i) **Byte Load** – for determining when to load the next byte into the **Parallel–to–Serial Converter**. (ii) **Word Load** – for determining when to load the next word into the **Shift Register Array**. (iii) **Space** – which is part of the **Start and Space Controller**, and ensures that the last two bytes of the preamble are transmitted as a continuous low level by deasserting the START–ENABLE signal.

The **Start and Space Controller** synchronizes the transmitter interface with the transmitter finite state controller. The START–ENABLE signal is asserted after the rising edge of the /LOAD–BYTE signal if both LOAD–WORD and TRANSMIT–ENABLE are asserted. The READY signal is asserted (by the finite state controller) whenever a new word is ready on the input data bus. The ACK signal is asserted after the rising edge of /LOAD–BYTE if LOAD–WORD is asserted (and the new word is loaded into the **Shift Register Array**). After the assertion of the ACK signal the READY signal is deasserted, and then ACK is also deasserted, to complete the handshake.

4.5.1. The operation of the transmitter link interface

The operation of the link interface has four basic phases: idle, start, running and termination, through which the **Shift Register Array**, **Parallel–to–Serial Converter**, **Byte Load Unary–Counter**, and **Word Load Unary–Counter** operate **continuously**. Only the **Start and Space Controller** changes its state during these four phases.

(1) *Idle phase* - TRANSMIT–ENABLE and START–ENABLE are deasserted. As a result, the **Serial–to–Parallel Converter** is loaded with zeros and the serial output is continuously low. The **Byte Load Unary–Counter** is loaded with 1 at the 12th position from the right, so that every 12 clock cycles (1500MHz) there is a /LOAD–BYTE pulse for one clock period. Every 8 /LOAD–BYTE pulses LOAD–WORD is asserted, and the **Shift Register Array** is loaded with 64–bit word from the **Data Bus**.

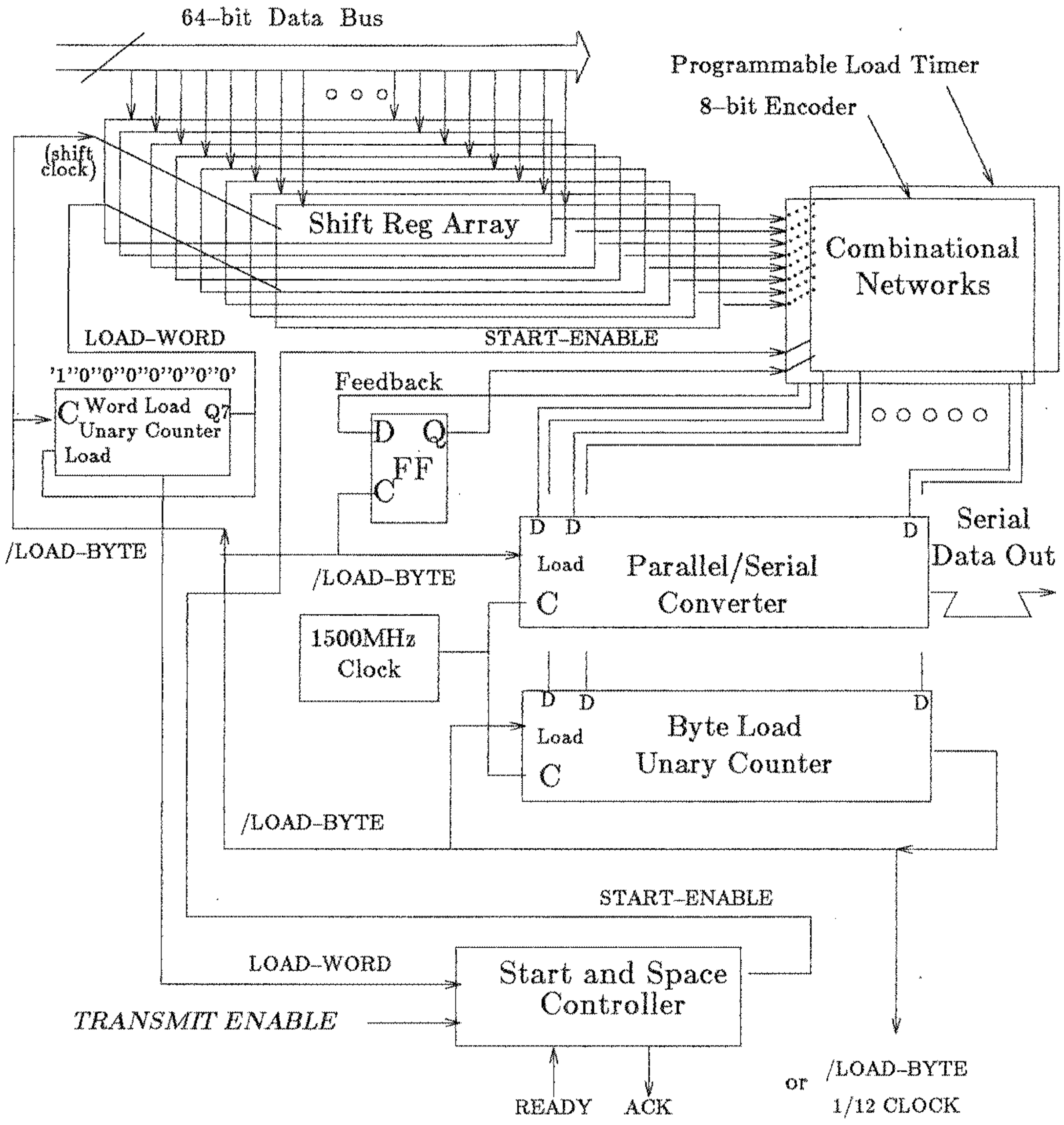

Figure 4.8: The Transmitter Link Level Interface

(2) *Start phase* - proceeds as follows: (1) TRANSMIT–ENABLE is asserted by the transmitter finite state controller. (2) START–ENABLE is asserted after the assertion of LOAD–WORD and /LOAD–BYTE. (3) Six bytes of the preamble are transmitted. (4) START–ENABLE is deasserted and two bytes of space are transmitted. (5) START–ENABLE is asserted and the first word of data is transmitted.

(3) *Running Phase* - data is transmitted continuously. (1) The ACK signal is asserted after the assertion of LOAD–WORD; the next word is loaded into the **Shift Register Array**. (2) The transmitter finite state controller deasserts READY. (3) The **Start and Space Controller** deasserts the ACK signal. (4) The transmitter finite state controller asserts the next data word onto the **Data Bus,** upon which READY is asserted. (5) Then the cycle repeats.

(4) *Termination phase* - after the last word is loaded, ACK causes the transmitter finite state controller to deassert both TRANSMIT–ENABLE and READY, and then ACK is deasserted. START–ENABLE is deasserted after the assertion of LOAD–WORD and the rising edge of the /LOAD–BYTE pulse.

4.6. Decoder and Serial–to–Parallel Converter

The basic principle of the interface operation, as previously explained, is to use the transition information of the incoming serial data both for decoding and for serial–to–parallel conversion. The receiver design for 8B/12B conservative encoding scheme is presented, and then some of the actual engineering issues are discussed.

4.6.1. The 8B/12B conservative decoder

A block diagram of the 8B/12B decoder for 1 gigabit/second is shown in Figure 4.9. The incoming serial data (Din) has exactly **three falling edges** for every 8 bits of data (or 12 bits of codeword). The third falling edge is the delimiter of the 12–bit codeword. Din is propagated via a transmission line (TL), which has 12 evenly spaced stubs as inputs to 12 flipflops. The Din signal also clocks the **Word Transition Unary–Counter.** This counter is designed for maximum bandwidth as described in Figure 4.7. The unary counter uses the falling edges of Din, and upon the third falling edge the /LOAD–BYTE signal is asserted, and then 12 bits are sampled from the TL into the 12 input flipflops.

The outputs of the 12 flipflops are decoded back to the original 8–bit data by the **Decoder Combinational Network**, and then fed into an array of 8 shift registers. There are two arrays of shift registers for double buffering, each containing a word of 64 bits. The signals CLK WORD1 and CLK WORD2 are used to clock the shift registers. Only one clock signal is active at a time, and they switch roles after 24 falling edges or after 8 rising edges of the /LOAD–BYTE signal (after 8 bytes of data are recorded). The word in the array which is not being clocked can be read by the receiver finite state controller into the input buffer. ACK and READY are handshake signals with the receiver finite state controller, and are used for synchronizing the transfers of words into the input buffer.

The **Space Detect** unit searches for the space between the preamble and the data. It uses a shift register with a 1500MHz clock for recording the incoming serial data into the shift register. A simple combinational network, which uses the shift

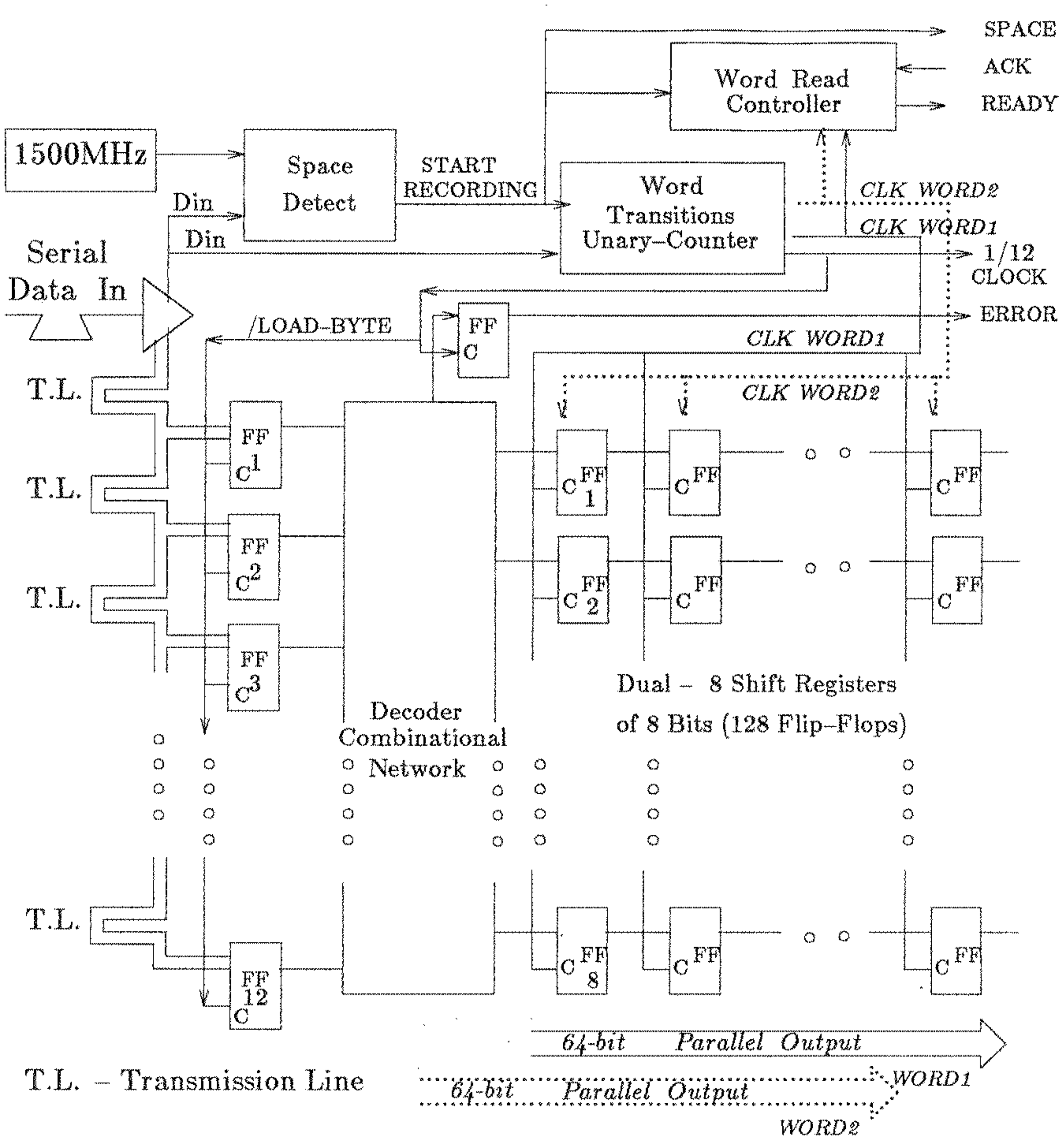

Figure 4.9: The 8B/12B Conservative Decoder

register outputs, is used to detect the space between the preamble and the body of the packet. This unit sends the signals START RECORDING or SPACE to the receiver finite state controller whenever the space is detected. This signal is also used for resetting the **Word Transitions Unary–Counter**.

Error detection is possible, since the codeword space is larger than the data word space (462 words vs. 256 words). If a codeword cannot be mapped to one of the data words, the ERROR signal is asserted. If a collision has occurred, the high levels of the Din signal are extended (transition information is destroyed), and with a high probability one of the succeeding input codewords is illegal.

The 1/12 CLOCK from the **Word Transitions Unary–Counter** is the clock signal to the receiver finite state controller, so that these two units are synchronized. While the input line is idle, the 1/12 CLOCK is derived from the local 1500MHz clock.

The 12 input flipflops can be realized on one integrated circuit, so they can be clocked simultaneously by the /LOAD–BYTE signal. The transmission line is realized on a printed circuit board (each of its segments is about 15–20 cm.).

4.6.2. Discussion

There are some engineering issues that affect the design, performance, and reliability of this interface. First, **group velocity dispersion** in the fiber causes pulse spreading $\Delta\tau$ proportional to the spectral width $\Delta\omega$ and the distance L:

$$\Delta\tau \approx \frac{2L}{v_g^2}\left|\frac{dv_g}{d\omega}\right|\Delta\omega$$

v_g is the group velocity, and $\Delta\tau$ determines the maximum bandwidth of the fiber. The pulse spreading distorts the serial signal shape, and makes the decoding and the serial–to–parallel conversion more difficult.

Another issue is the temperature stability of the TL, which determines the stability of the delays between the input flipflops. The PC board is thermally more stable than active devices, since it has a significant thermal inertia. The size and passive nature of the TL would suggest that it is much easier to stabilize at a constant temperature than either an active delay line or RC circuits.

It is important to note that the 1500MHz clock is not used for decoding or parallel–to–serial conversion. It is used only for control and error checking. The decoding and converting circuitry is completely asynchronous and can operate within the tolerances of the delay lines.

4.7. Conclusions

The previous discussion has shown the feasibility of constructing an interface that meets all its design objectives. The critical timing section of the transmitter is small enough to be implemented on a single GaAs gate–array, and the transmitting clock needs to drive only this chip. Similarly, the critical timing section of the decoder (the unary counter, input flipflops, and the shift/hold control) can be integrated on another single GaAs chip.

The use of unary counters (shift registers) is essential for achieving a maximum transmission rate in a given technology. They should be relatively small, in order to avoid clock skew delay, and the hierarchical design methodology makes this possible.

The new conservative encoding/decoding methodology is flexible, stable and simple enough to be used in a multiple--access medium. The decoder uses delays realized by transmission lines. This realization is simpler as the delay becomes shorter, and therefore favors higher baud rates (as long as the pulse dispersion is not significant).