US 20040022238A1

(19) **United States**

(12) **Patent Application Publication** (10) **Pub. No.: US 2004/0022238 A1**

Kimmitt (43) **Pub. Date: Feb. 5, 2004**

(54) **PHYSICAL CODING SUB-LAYER FOR TRANSMISSION OF DATA OVER MULTI-CHANNEL MEDIA**

(76) Inventor: **Myles Kimmitt**, Shrewsbury, MA (US)

Correspondence Address:
WEINGARTEN, SCHURGIN, GAGNEBIN & LEBOVICI LLP
TEN POST OFFICE SQUARE
BOSTON, MA 02109 (US)

(21) Appl. No.: **10/620,635**

(22) Filed: **Jul. 16, 2003**

Related U.S. Application Data

(63) Continuation of application No. 09/321,448, filed on May 27, 1999, now Pat. No. 6,618,395.

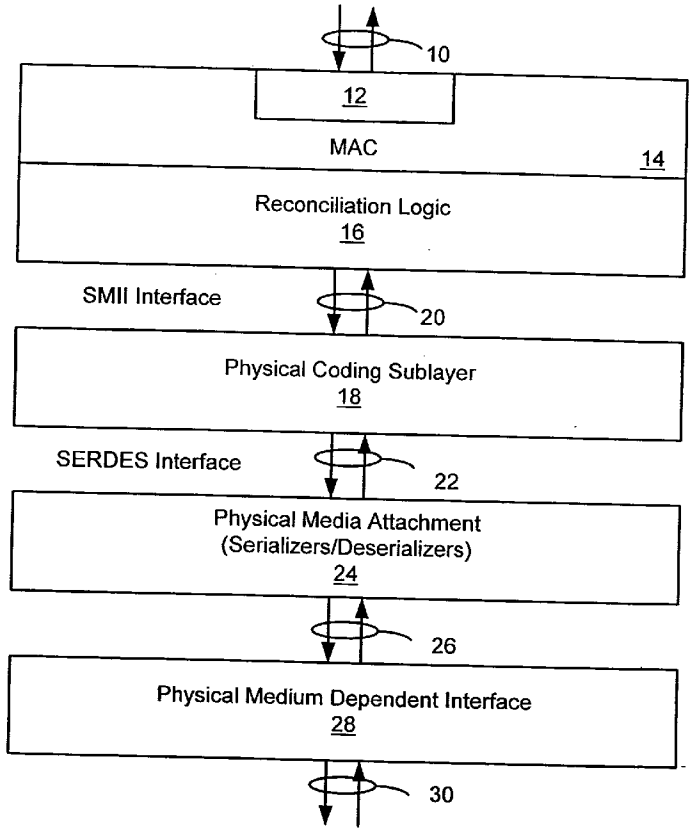
Publication Classification

(51) **Int. Cl.**[7] **H04Q 11/00**

(52) **U.S. Cl.** **370/366**; 370/537

(57) **ABSTRACT**

A method and apparatus for transporting data over a plurality of serial channels. A plurality of parallel data word are generated from a parallel data word of greater width. The plurality of parallel data words are scrambled in a predetermined manner utilizing a side scrambler to generate a plurality of cipher data words. A first bit is generated for each channel as an exclusive OR function from the cipher data word in the respective channel. A second bit is generated for each channel as an exclusive or function of the respective cipher data word and certain control information. The first bit is appended to the cipher data word for the channel from which it was derived. The second bit is appended to the cipher data word for a channel other than the one from which it was derived. The cipher data word and the first and second bits comprise a parallel extended width information word. The extended width information words are serialized and transmitted across a plurality of serial data channels corresponding in number to the number of parallel extended width information words. Receive logic is provided for each serial channel which converts received serial data to parallel data, obtain word synchronization and achieves aligns the words received over the respective channels to assure avoid word skew misalignment across channels. The received data is descrambled and recombined in the receive logic to obtain the originally transmitted data word. Offset side scramblers are designed so as to reduce near end and far end crosstalk.



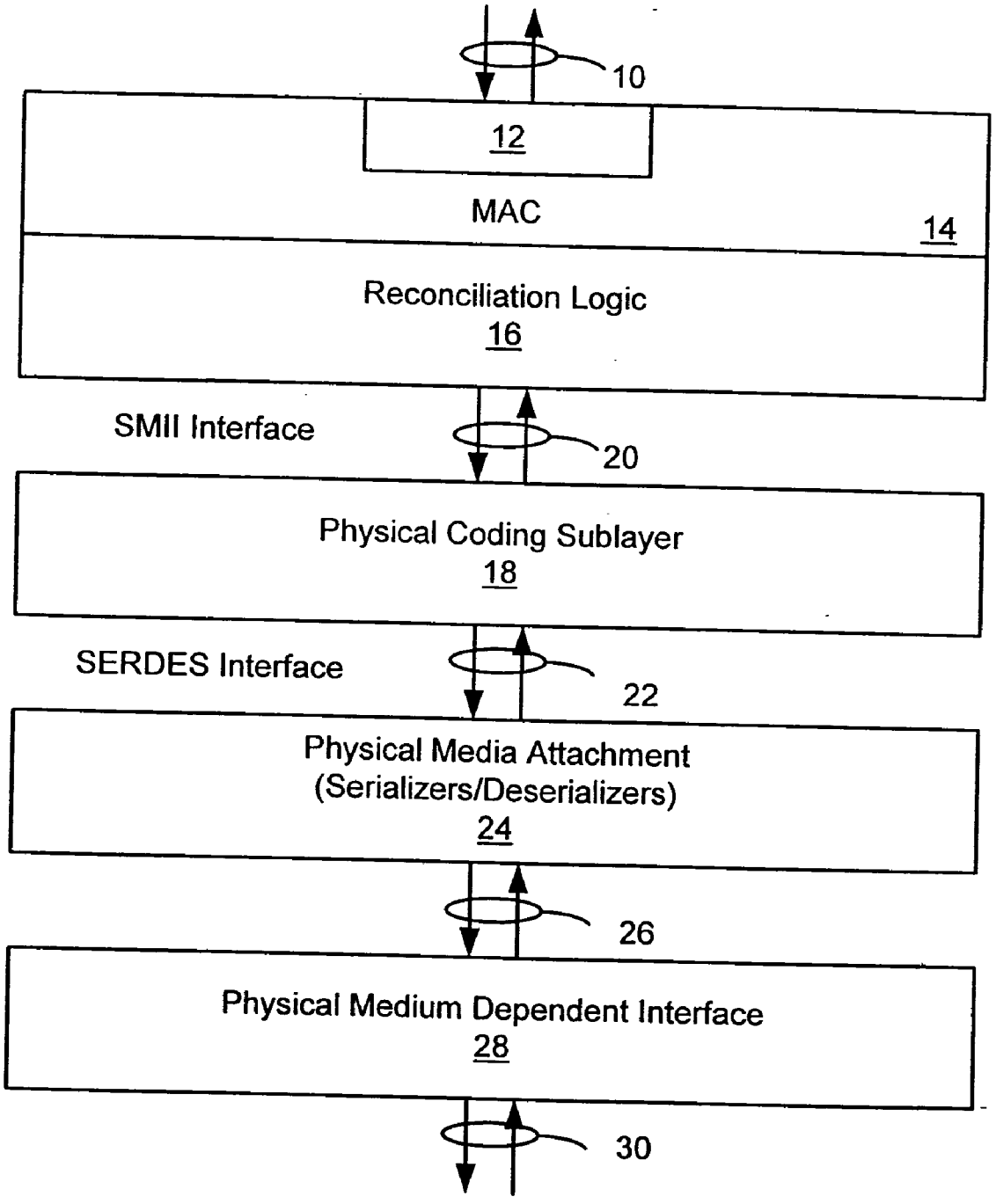

10
12
MAC
14
Reconciliation Logic
16
SMII Interface
20
Physical Coding Sublayer
18
SERDES Interface
22
Physical Media Attachment
(Serializers/Deserializers)
24
26
Physical Medium Dependent Interface
28
30

Figure 1

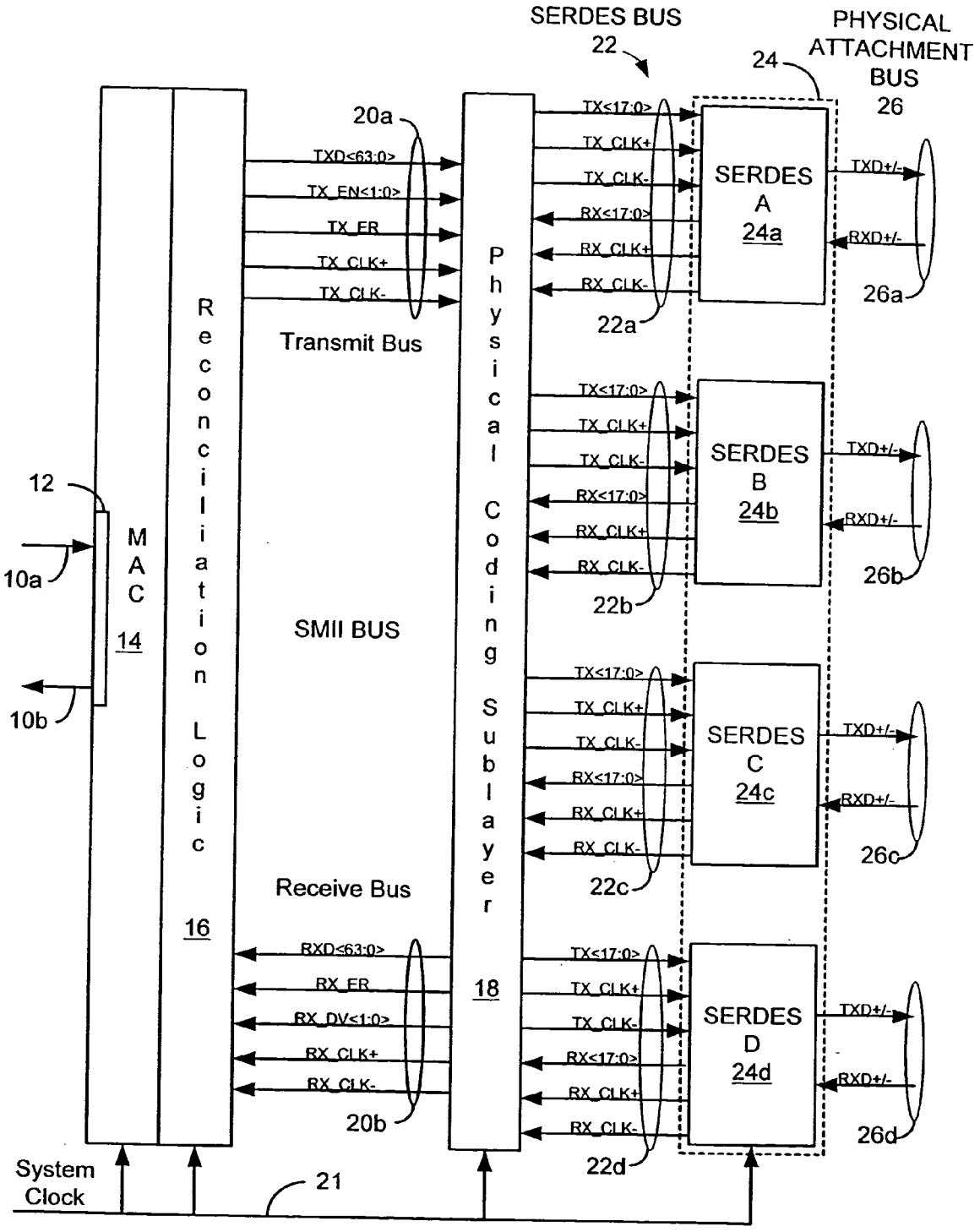

SERDES BUS
22
PHYSICAL ATTACHMENT BUS
24
26
20a
TXD<63:0>
TX_EN<1:0>
TX_ER
TX_CLK+
TX_CLK-
Transmit Bus
TX<17:0>
TX_CLK+
TX_CLK-
RX<17:0>
RX_CLK+
RX_CLK-
22a
SERDES A 24a
TXD+/-
RXD+/-
26a
SERDES B 24b
22b
26b
SERDES C 24c
22c
26c
SERDES D 24d
22d
26d
12
10a
10b
MAC 14
Reconciliation Logic 16
SMII BUS
Receive Bus
RXD<63:0>
RX_ER
RX_DV<1:0>
RX_CLK+
RX_CLK-
20b
Physical Coding Sublayer 18
System Clock
21

Figure 2

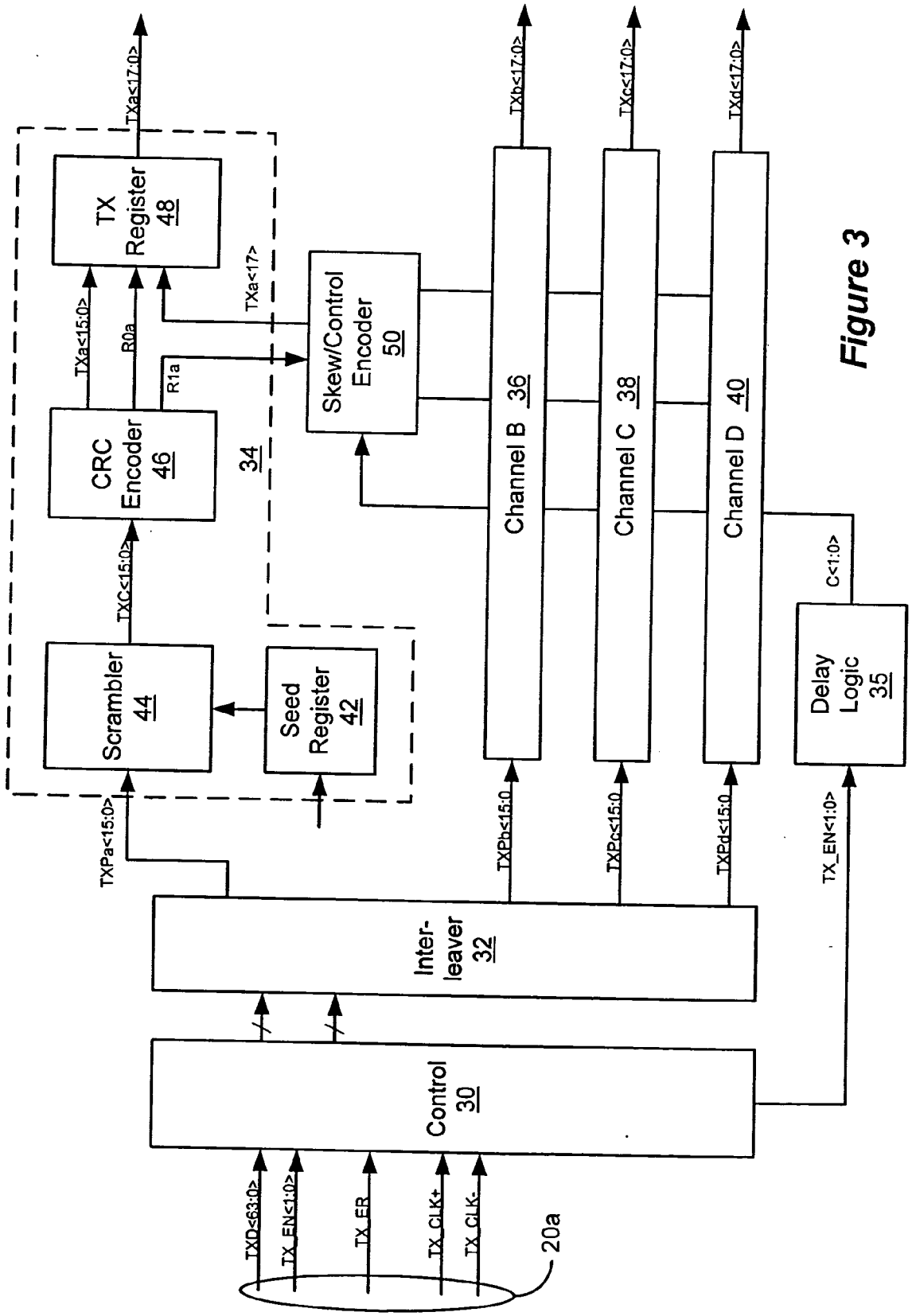

TXa<17:0>
TX Register 48
TXa<17>
Skew/Control Encoder 50
TXb<17:0>
TXc<17:0>
TXd<17:0>
Channel B 36
Channel C 38
Channel D 40
TXa<15:0>
R0a
R1a
CRC Encoder 46
34
TXC<15:0>
Scrambler 44
Seed Register 42
C<1:0>
Delay Logic 35
TXPa<15:0>
TXPb<15:0
TXPc<15:0
TXPd<15:0
TX_EN<1:0>
Inter-leaver 32
Control 30
TXD<63:0>
TX_EN<1:0>
TX_ER
TX_CLK+
TX_CLK-
20a

Figure 3

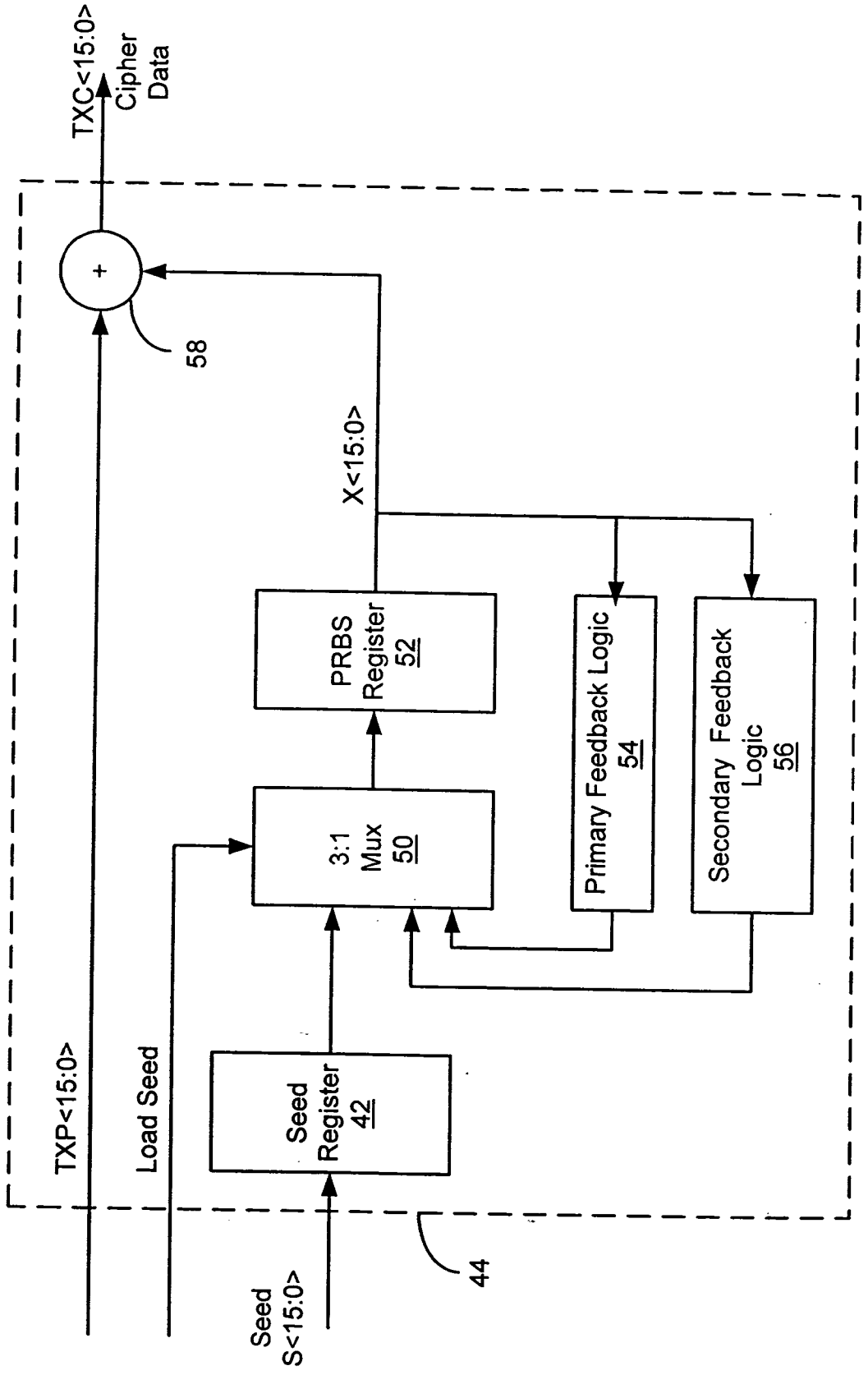

TXC<15:0>
Cipher Data
+
58
X<15:0>
PRBS Register 52
3:1 Mux 50
Primary Feedback Logic 54
Secondary Feedback Logic 56
Seed Register 42
TXP<15:0>
Load Seed
Seed S<15:0>
44

Figure 4

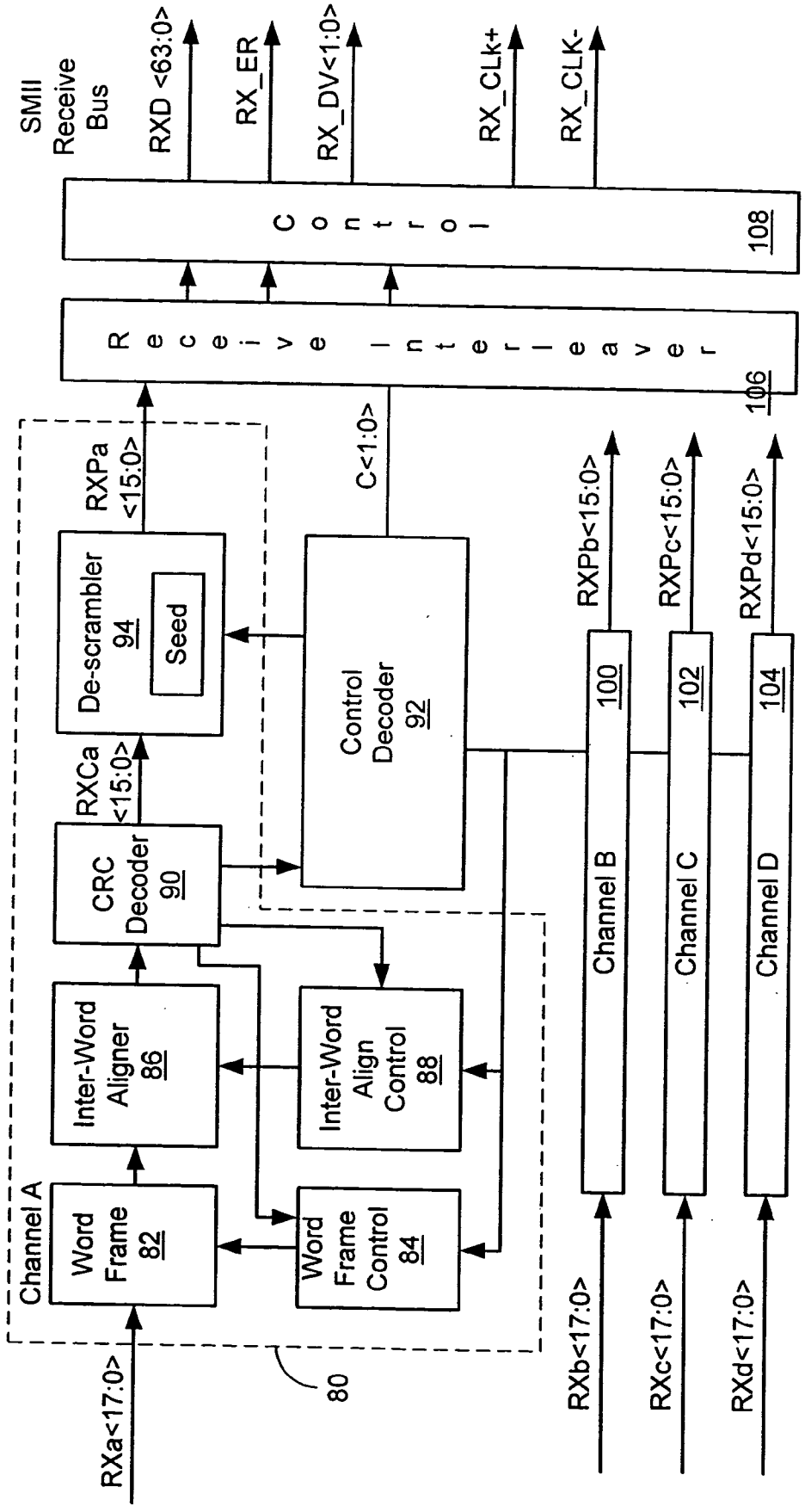

SMII Receive Bus
RXD <63:0>
RX_ER
RX_DV<1:0>
RX_CLk+
RX_CLK-
Control
108
Receive Interleaver
106
C<1:0>
RXPa <15:0>
RXPb<15:0>
RXPc<15:0>
RXPd<15:0>
De-scrambler 94
Seed
Control Decoder 92
RXCa <15:0>
CRC Decoder 90
Inter-Word Aligner 86
Inter-Word Align Control 88
Word Frame 82
Word Frame Control 84
Channel A
Channel B 100
Channel C 102
Channel D 104
RXa<17:0>
RXb<17:0>
RXc<17:0>
RXd<17:0>
80

Figure 5

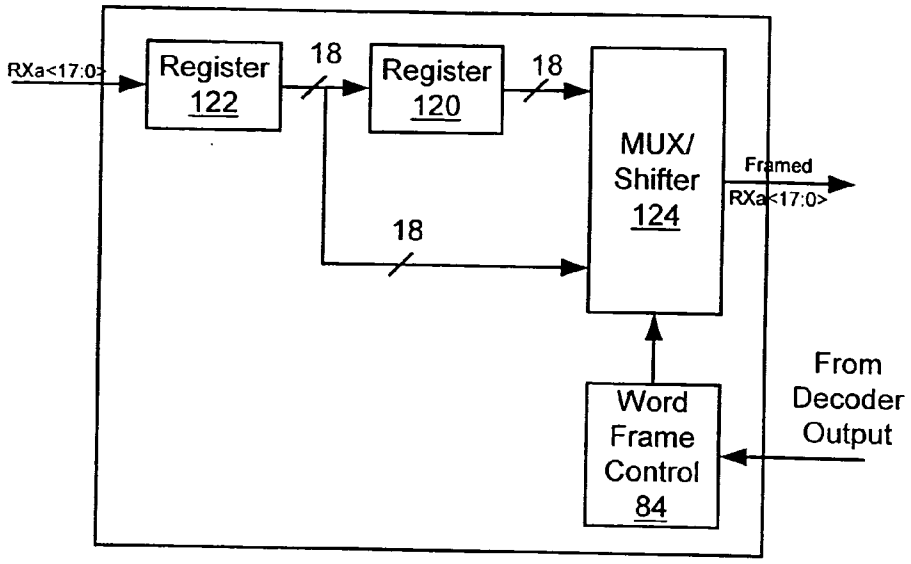


Figure 6

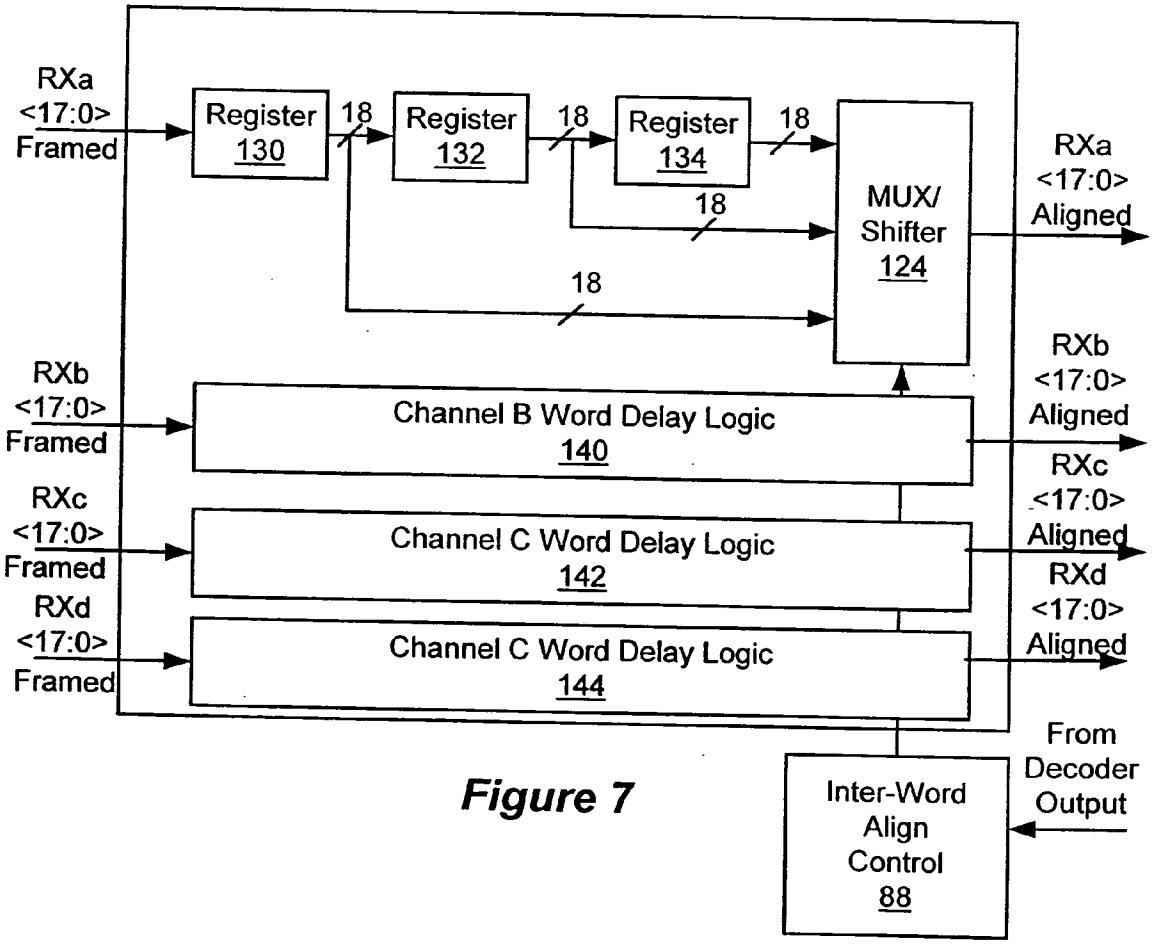


Figure 7

PHYSICAL CODING SUB-LAYER FOR TRANSMISSION OF DATA OVER MULTI-CHANNEL MEDIA

CROSS REFERENCE TO RELATED APPLICATIONS

[0001] This application is a continuation application of U.S. patent application Ser. No. 09/321,448 filed May 27, 1999.

STATEMENT REGARDING FEDERALLY SPONSORED RESEARCH OR DEVELOPMENT

[0002] N/A

BACKGROUND OF THE INVENTION

[0003] The present invention relates generally to data encoding and telecommunications and more specifically, to a physical encoding and decoding sub-layer operative to permit the reliable communication of data across a plurality of serial channels.

[0004] In recent years there has been an increasing desire to produce electronic products which operate at ever increasing speeds. Of particular note in this regard are telecommunications devices such as routers, bridges and switches. While typical communication line rates for such devices were 10 megabits per second (mbps) for ethernet transmissions less than a decade ago, 100 mbps ethernet line rates have now become commonplace. Moreover, devices are currently being deployed which support 1 gigabit per second (gbps) ethernet line rates.

[0005] Data is typically received at an input port of a telecommunications device over a high speed serial communications link. Received data is converted to a parallel word format in accordance with a specified media access control (MAC) for processing within the device. The width of the parallel data output from the MAC protocol is typically specified for the respective MAC protocol. A 10 gb ethernet protocol has been described having a MAC output in the form of a 64 bit wide data words. At such high data rates, the transport of data within the device can be problematic.

[0006] While data can be transported through the device as a parallel word to achieve workable clock rates, such is undesirable for a number of reasons. First, wide bus widths consume substantial space on the printed circuit boards for the numerous conductive paths which are required. Second, passing large numbers of conductive signals through backplanes requires large numbers of connector contacts. Often, it is undesirable to provide for the large number of connector contacts that are required to accommodate a 64 bit wide or greater width parallel bus. Additionally, it is recognized that interconnections through backplane connectors contribute to system unreliability and for this reason as well it is preferable to minimize the number of signal paths through backplanes and connectors. Finally, numerous integrated circuits are required in terms of drivers and receivers to interface to wide parallel buses.

[0007] In order to minimize the number of printed circuit board runs and backplane connections, parallel data has been segregated into narrower parallel data words and the respective words have been serialized for transmission over a plurality of serial channels. Complex techniques and/or substantial overhead in terms of signal bandwidth and circuitry have typically been necessary to accomplish synchronization and/or framing in such systems.

[0008] Moreover, it is desirable to be able for the receive logic to be able to be acquire synchronization of serially transmitted data "blind"; i.e. without the use of additional synchronization signal line since the transport of separate synchronization signals also adds to the number of printed circuit board runs and connector contacts that are needed.

[0009] It would therefore be desirable to have a data coding, decoding and transport technique which allows the transmission of data reliably through a telecommunications device without employing wide parallel buses and which permits blind acquisition of synchronization at the receiver and reassembly of the transmitted data words.

BRIEF SUMMARY OF THE INVENTION

[0010] In accordance with the present invention, a method and apparatus for transporting data over a plurality of serial channels is disclosed. A wide parallel data word is subdivided into a plurality of lesser width parallel data words in which the bits from the wide data word are interleaved across the lesser width data words. The lesser width data words are each XORed in parallel with the output of a side scrambler to decorrelate the data transmitted within each channel. The outputs of the side scrambler for each channel comprise cipher data which is decorrelated from the remaining channels to reduce near end crosstalk (NEXT) between channels. The cipher data is applied to a CRC encoder which, in the present embodiment, generates two additional bits. The two additional bits are derived from the cipher data for the respective channels. The first bit is transmitted in the same channel as the cipher data from which it was derived and is used to verify word alignment at the receive end of the respective serial channel. The second bit is used to assure inter-channel skew. The second bit is rotated across the channel, i.e. transmitted in a channel other than one containing the cipher data from which it was derived. Additionally, control information is encoded on the second bit for each of the respective channels. The control information indicates whether the data word (a) contains an idle signal, (b) contains data valid in the low order bytes, (c) contains data valid in the high order bytes, or (d) contains valid data across all of the bytes.

[0011] The cipher data plus the two additional bits are serialized for each channel and transmitted serially over a serial link or channel. The serial data transmitted over each channel is then deserialized within receive logic at the receive end of the respective serial channel.

[0012] At the receive end of the link, logic is provided to permit the acquisition of word frame alignment, to permit acquisition of inter-channel skew alignment and to obtain the proper seed for use by a descrambler associated with each channel so the cipher data can be converted back into the actual data after synchronization has been achieved. The synchronization process involves a number of steps.

[0013] First, word frame alignment is achieved within each channel using the first of the two additional bits transmitted over the serial link. Next, the second bit is employed to achieve inter-channel skew alignment. Once

inter-channel skew alignment is achieved, the receiver can identify idle symbols transmitted over the respective link from the control information encoded on the second bits of each channel. The idle information is then XORed with the cipher data to acquire the seed for each of the channels. The parallel cipher data is then XORed in the receiver for each channel with the output of the descrambler for the respective channel using the seed acquired for each channel. The output of the descrambler corresponds to the actual data transmitted over the respective serial links prior the scrambling of such data in the transmitter. Finally, the parallel data output for each channel in the receiver is combined to form the original wide parallel data word.

[0014] In a preferred embodiment, a primary pseudo-random sequence generator and a secondary pseudo-random number generator is replicated in each of the scramblers and descramblers. To reduce far end crosstalk (FEXT) between receiver and transmitter signals, the transmission of data in one direction over the plurality of channels proceeds using the primary pseudo-random sequence generators and transmission of data in the opposite direction proceeds using the secondary pseudo-random sequence generators in the transmitter scramblers and the receiver descramblers. The primary and secondary pseudo-random sequence generators are selected so as to decorrelate data transmitted by transmitters in opposite directions over the serial channels.

BRIEF DESCRIPTION OF THE SEVERAL VIEWS OF THE DRAWING

[0015] The invention will be more fully understood by reference to the following Detailed Description of the Invention in conjunction with the drawing of which:

[0016] **FIG. 1** is a block diagram illustrating a physical coding sublayer within a system operative in accordance with the present invention;

[0017] **FIG. 2** is a block diagram illustrating bus signals employed in the system depicted in **FIG. 1**;

[0018] **FIG. 3** is a block diagram illustrating the transmit portion of the physical coding sublayer depicted in **FIGS. 1 and 2**;

[0019] **FIG. 4** is a block diagram of a side scrambler for scrambling data in each channel of the transmit portion of the physical coding sublayer;

[0020] **FIG. 5** is a block diagram illustrating the receive logic associated with the physical coding sublayer depicted in **FIGS. 1 and 2**;

[0021] **FIG. 6** is a block diagram illustrating word framing logic; and

[0022] **FIG. 7** is a block diagram illustrating skew alignment logic.

DETAILED DESCRIPTION OF THE INVENTION

[0023] In accordance with the present invention a physical coding sublayer is disclosed which facilitates communication across multiple serial channels in short haul copper links (such as backplanes and short cable connections) and wave division multiplexing (WDM) applications (employing fiber optic media). A system incorporating the presently disclosed physical coding sublayer is illustrated in **FIG. 1**. Referring to **FIG. 1**, data is received from a host system interface over a communication link **10** at a system interface **12**. The system interface is coupled to media access control (MAC) circuitry **14** and provides the interface between the host system interface and the MAC **14**. The MAC **14** controls the reception and transmission of data from and to the physical layer in accordance with techniques well known in the art.

[0024] In the embodiment herein described, a physical coding sublayer is illustrated which supports a 10 gigabit per second (gbps) ethernet MAC. It should be noted, however, that the techniques and methods explained herein are scaleable and may be employed to support MACs having different parallel output word widths and different line rates.

[0025] Data and control signals provided at the output of the MAC **14** are coupled to reconciliation logic **16**. The reconciliation logic **16** is coupled to the physical coding sublayer logic **18** via the scaleable media independent interface (SMII) bus interface **20**. The reconciliation logic **16** serves to provide any necessary conversion between the input/output signals associated with the MAC and the signals which make up the SMII bus **20**. Data is conveyed between the reconciliation logic **16** and the physical coding sublayer (PCS) **18** via transmit and receive buses, each of which comprises a wide parallel data bus. The physical coding sublayer segments the wide parallel data words received from the reconciliation logic **16** into a plurality of narrower parallel data words and encodes the respective narrower data words as hereinafter described in greater detail to form a corresponding plurality of encoded parallel data words. The encoded parallel data words are conveyed over a Serializer/Deserializer (SERDES) Interface bus **22** from the PCS **18** to a plurality of serializers within Serializer/Deserializer (SERDES) logic **24**. The SERDES logic **24** converts the received encoded parallel data words into serial encoded data. A physical attachment bus **26** couples the SERDES logic **24** to the driver logic **28** for the applicable media. More specifically, the physical attachment bus **26** includes a plurality of serial data channels for carrying serialized encoded data from the SERDES logic **24** to the driver logic **28** or, in the reverse direction for carrying serial data from the driver logic **28** to the SERDES logic **24**.

[0026] The driver logic **28** provides the appropriate electrical or electro-optical interface to a copper or fiber media, as applicable. By subdividing the wide parallel data word into a plurality of narrower parallel data words, encoding the narrower parallel data words and transmitting data contained in the plurality of narrower encoded parallel data words serially over a corresponding plurality data of serial channels, the number of printed circuit board runs are greatly reduced, significant space savings on the printed circuit board may be achieved and fewer connector contacts are needed.

[0027] **FIG. 2** illustrates the scalable media independent interface (SMII) **20**, the SERDES interface **22** and the physical attachment interface **26** in greater detail. The SMII bus **20** and the SERDES interface bus **22** each include parallel data buses and control signals. The functions of the respective signals in each of the buses are described below. The physical attachment bus **26**, in the illustrated embodiment, includes four channels identified as channels **26*a*-26*d***

respectively. Each channel includes two differential signalling pairs. One of the differential signalling pairs in each channel is employed for transmission of serial data and the other one of the differential signalling pairs in each channel is employed for reception of serial data.

[0028] The Scaleable Media Independent Interface (SMII) bus **20** is used to connect the reconciliation logic **16** with the PCS layer **18**. In the illustrated embodiment, the data width of the parallel data word output from the MAC is 64 bits as is the width of the data word output from the reconciliation logic **16**. The SMII bus **20** includes control lines (TX_EN<1:0>) which include control information that is employed to identify which bytes are idle characters or contain valid data when the full 64 bits are not valid.

[0029] The 64-bit bus is the standard width for the 10 GB/s encoding system. Standardization of the SMII bus interface is useful since other 10 GB/s physical interconnect technologies may be developed (such as a full rate optical interface) and accordingly, the physical coding sublayer will be reusable in different interconnect systems.

[0030] The SMII bus **20** includes a transmit bus **20***a* comprising data and control signals driven by the reconciliation logic **16** and destined for the physical coding sublayer **18**. Additionally, the SMII bus **20** includes a receive bus **20***b* comprising data and control signals driven by the physical coding sublayer **18** and destined for the reconciliation logic **16**.

[0031] The Transmit Bus

[0032] The transmit bus **20***a* includes sixty four transmit data lines TXD <63:0>, two transmit enable lines TX_EN<1:0>, an optional TX_ER line, and TX_CLK+ and TX_CLK− lines. The functions and use of the respective transmit bus signals are described below.

[0033] Transmit Data, TXD<63:0>

[0034] The Transmit Data Bus (TXD) is used to transfer transmit data and control from the MAC reconciliation logic **16** to the PCS **18**. TXD<63> is defined as the most significant bit and TXD<0> is defined as the least significant bit. Byte order is shown in Table 1 and encoding is shown in Tables 2 through 4.

TABLE 1

Transmit Data Bus Byte Order

TXD <63:56>	TXD <55:48>	TXD <47:40>	TXD <39:32>	TXD <31:24>	TXD <23:16>	TXD <15:8>	TXD <7:0>
Byte 7	Byte 6	Byte 5	Byte 4	Byte 3	Byte 2	Byte 1	Byte 0

[0035] Transmit Enable, TX_EN<1:0>

[0036] The Transmit Enable signals indicate the presence of data on the Transmit Data Bus. Two signals are employed to define bytes of valid data in either the upper or lower bytes of the Transmit Data Bus. This is required for Ethernet frames and the inter packet gap since it cannot be assured that valid data will fall on 8-byte boundaries. Four Transmit Enable states are defined: Idle (no data transmission), Data Valid All, Data Valid Low, Data Valid High. Encoding of the Transmit Enable states is shown in Tables 2 through 4.

[0037] Transmit Error, TX_ER

[0038] Transmit Error is an optional control signal that is used by the MAC reconciliation logic **16** to indicate an error to the receiving MAC with a high degree of confidence. Transmit Error can occur at any time during normal frame transmission (i.e. when Transmit Enable is active) and the PCS is responsible for signaling an error within that frame, although the position of the error within the frame is not defined.

[0039] Transmit Clock, TX_CLK<P:N>

[0040] TX_CLK is a continuous differential transmit clock, generated by the MAC reconciliation **16**, and used to transfer TXD, RX_DV<1:0>and TX_ER to the PCS **18**. Sampling occurs on the rising edge of TX_CLK+. This clock is nominally running at 156.25 MHz in a preferred embodiment of the 64-bit wide implementation.

[0041] Transmit Signal Encoding

[0042] TX_EN<1:0>and TX_ER signals are used to define whether idles are being carried on the the transmit data lines TXD<63:0>as shown in Table 2 and which data within the full width data word is valid. In the idle state, the transmit medium is not actively transmitting data. The three data valid states indicate which bytes of data are valid. Three undefined transmit error states are also provided which may be employed for other purposes.

TABLE 2

Transmit Data Bus Control Encoding

TX_EN 1	TX_EN 0	TX_ER	TXD<63:0>	Description
0	0	0	0x5555555555555555	Idle
0	0	1	Reserved	Idle on medium, reserved communication
1	1	0	0x00..00 to 0xFF..FF	Transmit Data Valid All
0	1	0	See Table 3	Transmit Data Valid Low
1	0	0	See Table 4	Transmit Data Valid High
1	1	1	Don't Care	Transmit Error

TABLE 2-continued

Transmit Data Bus Control Encoding

TX_EN 1	TX_EN 0	TX_ER	TXD<63:0>	Description
0	1	1	Don't Care	Transmit Error
1	0	1	Don't Care	Transmit Error

[0043] Further encoding of valid data bytes is contained within the data bus itself during the Data Valid High and Data Valid Low states. These are shown in Tables 3 and Table 4.

TABLE 3

Transmit Data Valid Low Encoding

TXD <63:59>	TXD <58>	TXD <57>	TXD <56>	TXD <55:0>	Description
Reserved	0	0	0	Reserved	Reserved
Reserved	0	0	1	Idle*/Data	1 byte valid, TXD<7:0>
Reserved	0	1	0	Idle*/Data	2 bytes valid, TXD<15:0>
Reserved	0	1	1	Idle*/Data	3 bytes valid, TXD<23:0>
Reserved	1	0	0	Idle*/Data	4 bytes valid, TXD<31:0>
Reserved	1	0	1	Idle*/Data	5 bytes valid, TXD<39:0>
Reserved	1	1	0	Idle*/Data	6 bytes valid, TXD<47:0>
Reserved	1	1	1	Idle*/Data	7 bytes valid, TXD<55:0>

*Unused TXD bytes are set to 0x55

[0044]

TABLE 4

Transmit Data Valid High Encoding

TXD <7:3>	TXD <2>	TXD <1>	TXD <0>	TXD <55:0>	Description
Reserved	0	0	0	Reserved	Reserved
Reserved	0	0	1	Data/Idle*	1 byte valid, TXD<63:56>
Reserved	0	1	0	Data/Idle*	2 bytes valid, TXD<63:48>
Reserved	0	1	1	Data/Idle*	3 bytes valid, TXD<63:40>
Reserved	1	0	0	Data/Idle*	4 bytes valid, TXD<63:32>
Reserved	1	0	1	Data/Idle*	5 bytes valid, TXD<63:24>
Reserved	1	1	0	Data/Idle*	6 bytes valid, TXD<63:16>
Reserved	1	1	1	Data/Idle*	7 bytes valid, TXD<63:8>

*Unused TXD bytes are set to 0x55

[0045] The Receive Bus

[0046] The receive bus is employed for data forwarding from the physical coding sublayer **18** to the reconciliation logic **16** and includes sixty four receive data lines RXD<63:0>, two RX_DV<0:1>lines, an RX_ER line, and clock lines RX_CLK+ and RX_CLK−. The function of each of the receive bus signal lines is described below.

[0047] Receive Data, RXD<63:0>

[0048] The Receive Data Bus (RXD<63:0>) is used to transfer receive data and control information from the PCS layer **18** to the MAC reconciliation logic **16**. RXD<63>is defined as the most significant bit and RXD<0>is defined as the least significant bit. Byte order is shown in Table 5 and encoding is shown in Tables 6 through 8.

TABLE 5

Receive Data Bus Byte Order

RXD <63:56>	RXD <55:48>	RXD <47:40>	RXD <39:32>	RXD <31:24>	RXD <23:16>	RXD <15:8>	RXD <7:0>
Byte 7	Byte 6	Byte 5	Byte 4	Byte 3	Byte 2	Byte 1	Byte 0

[0049] Receive Data Valid, RX_DV<1:0>

[0050] The receive data valid signals RX_DV<1:0>indicate the presence of data on the receive data bus lines RXD<63:0>. The signals RX_DV<1:0>are provided to identify the bytes of valid data on the receive bus RXD<63:0>. This facility is necessary since the length of Ethernet frames and inter-packet gaps do not necessarily fall on 8-byte boundaries. Four receive states are possible: Idle (no data reception), receive data valid all, receive data valid low, receive data valid high. The encoding of Receive Data Valid signals is shown in Tables 6 through 8.

[0051] Receive Error, RX_ER

[0052] Receive Error RX_ER is used by the PCS layer **18** to signal the MAC reconciliation logic **16** that an error has been detected in the received frame. Receive Error can occur at any time during normal frame reception; i.e. when Receive Data Valid is active and the PCS **18** is responsible for signaling an error within that frame, although the position of the error within the frame is not defined. Use of the Receive Error signal during idle is reserved for in-band communication between the MAC **14** and the PCS layer **18**.

[0053] Receive Clock, RX_CLK<P:N>

[0054] RX_CLK+ and RX_CLK− comprise a continuous differential receive clock, generated by the PCS layer **18**, and is used to transfer RXD, RX_DV<1:0>and RX_ER to the MAC reconciliation logic **16**. In a preferred embodiment, sampling occurs on the rising edge of RX_CLK+. The clock is nominally running at 156.25 MHz in the illustrated implementation.

[0055] The RX clock is derived from the received data streams during normal receive operation using a phase locked loop or any other suitable clock recovery technique. The PCS **18** uses continuous signaling on the medium, i.e. idle characters are transmitted over the receive bus during periods when actual data is not being transmitted. The RX clock is derived from a local reference clock (such as a reference or TX clock) when receive streams are not present. Transitions between the recovered clock and the local reference clock are accomplished by extending either the high or low state of the RX clock.

[0056] Receive Signal Encoding

[0057] RX_DV<1:0>and RX_ER define primary control of the Receive Data bus as shown in Table 6. An idle state is provided for transmission of idle characters when the receive data bus is not carrying active data. Three data valid states indicate which bytes of receive data are valid. The data valid states are valid when the RX_ER signal is deasserted.

TABLE 6

Receive Data Bus Control Encoding

RX_DV1	RX_DV0	RX_ER	RXD<63:0>	Description
0	0	0	0x5555555555555555	Idle
0	0	1	Reserved	Idle on medium, reserved communication
1	1	0	0x00..00 to 0xFF..FF	Receive Data Valid All
0	1	0	See Table 7	Receive Data Valid Low
1	0	0	See Table 8	Receive Data Valid High
1	1	1	Don't Care	Receive Error
0	1	1	Don't Care	Receive Error
1	0	1	Don't Care	Receive Error

[0058] Further encoding of valid data bytes is contained within the Receive Data bus (RXD data lines) during times when the Data Valid High and Data Valid Low states are active. These are shown in Table 7 and Table 8.

TABLE 7

Receive Data Valid Low Encoding

RXD <63:59>	RXD <58>	RXD <57>	RXD <56>	RXD <55:0>	Description
Reserved	0	0	0	Reserved	Reserved
Reserved	0	0	1	Idle*/Data	1 byte valid, RXD<7:0>
Reserved	0	1	0	Idle*/Data	2 bytes valid, RXD<15:0>
Reserved	0	1	1	Idle*/Data	3 bytes valid, RXD<23:0>
Reserved	1	0	0	Idle*/Data	4 bytes valid, RXD<31:0>
Reserved	1	0	1	ldle*/Data	5 bytes valid, RXD<39:0>
Reserved	1	1	0	Idle*/Data	6 bytes valid, RXD<47:0>
Reserved	1	1	1	Idle*/Data	7 bytes valid, RXD<55:0>

*Unused RXD bytes are set to 0x55

[0059]

TABLE 8

Receive Data Valid High Encoding

RXD <7:3>	RXD <2>	RXD <1>	RXD <0>	RXD <55:0>	Description
Reserved	0	0	0	Reserved	Reserved
Reserved	0	0	1	Data/Idle*	1 byte valid, RXD<63:56>
Reserved	0	1	0	Data/Idle*	2 bytes valid, RXD<63:48>
Reserved	0	1	1	Data/Idle*	3 bytes valid, RXD<63:40>
Reserved	1	0	0	Data/Idle*	4 bytes valid, RXD<63:32>
Reserved	1	0	1	Data/Idle*	5 bytes valid, RXD<63:24>
Reserved	1	1	0	Data/Idle*	6 bytes valid, RXD<63:16>
Reserved	1	1	1	Data/Idle*	7 bytes valid, RXD<63:8>

*Unused RXD bytes are set to 0x55

[0060] As illustrated in **FIG. 2**, the SERDES interface bus **22** includes four groups of data and control lines **22*a*-22*dd*** corresponding to data and control signals associated with each of four channels. Each one of the four channels on the SERDES interface includes 18 data lines TX<17:0>, two transmit clock lines TX_CLK+ and TX_CLK−, 18 receive data lines RX<17:0>and two receive clock lines RX_CLK+ and RX_CLK−.

[0061] Parallel data received over the transmit buses (TX bus) by the serializers within SERDES logic **24*a*-24*d*** respectively, is serialized for transmission over the differential serial transmit link (TXD+/−) for the corresponding channel **26*a*-26*d*** of the physical attachment bus **26**. In the present embodiment, each channel **26*a*-26*d*** includes a differential serial data link TXD+/− for data driven by the respective serializer within the SERDES logic **24** and a serial data link RXD+/− for data destined for the respective deserializer within the SERDES logic **24**. While the disclosed embodiment employs differential data and clock signals, single ended drivers and receivers may be employed.

[0062] The transmit path through the physical coding sublayer **18** is illustrated in greater detail in **FIG. 3**. The transmit path includes a control block **30** which is operative to decode the SMII transmit state and to control the data path contents during control symbols. Inband communications from the MAC during the inter packet gap may be decoded and separated in the control block.

[0063] The transmit interleave logic **32** segments the 64 bit wide data word received from the 10 gbps MAC into four sixteen-bit wide data words. The bits are interleaved into the channels to give better error detection properties to the MAC CRC. More specificically, by interleaving the bits from the 64 bit wide data word across the narrower sixteen bit wide data words such that each successive bit of the wide data word is contained within a different one of the sixteen bit wide data words, burst error on any of the channels will appear as dispersed errors in the MAC data stream . Such increases the probability of error detection. Shown below are the equations which illustrate how the parallel data input within the 64 bit wide parallel data word is interleaved across and segmented into the four sixteen bit data words in the present embodiment.

[0064] TXPa<15:0>=<TXD60, TXD56, TXD52, TXD48, TXD44, TXD40, TXD36, TXD32, TXD28, TXD24, TXD20, TXD16, TXD12, TXD8, TXD4, TXD0>

[0065] TXPb<15:0>=<TXD61, TXD57, TXD53, TXD49, TXD45, TXD41, TXD37, TXD33, TXD29, TXD25, TXD21, TXD17, TXD13, TXD9, TXD5, TXD1>

[0066] TXPc<15:0>=<TXD62, TXD58, TXD54, TXD50, TXD46, TXD42, TXD38, TXD34, TXD30, TXD26, TXD22, TXD18, TXD14, TXD10, TXD6, TXD2>

[0067] TXPd<15:0>=<TXD63, TXD59, TXD55, TXD51, TXD47, TXD43, TXD39, TXD35, TXD31, TXD27, TXD23, TXD 19, TXD 15, TXD 11, TXD7, TXD3>

[0068] The output from the interleave logic **32** thus comprises four 16 bit parallel data words which are coupled to channel logic **34**, **36**, **38**, **40** via buses TXPa through TXPd.

[0069] The channel logic for each of the four channels includes scrambler logic **44**, CRC encoder logic **46** and a transmit register **48**. While the channel logic is depicted in **FIG. 3** only for channel A for ease of illustration, the same logic is replicated in the remaining channels.

[0070] Scrambler Operation

[0071] The side scrambler **44** is operative to provide DC balance, provide clock transitions to allow clock recovery at the receiver, and to spread the signal spectrum across the available bandwidth and minimize EMI. The incoming data to the scrambler logic **44** is carried on the respective TXP bus and is driven by the transmit interleave logic **32**. The incoming data received over the TXP bus is XORED with a sixteen bit wide output from a pseudo-random binary sequence generator as hereinafter discussed in greater detail.

[0072] Each of the channels **34**, **36**, **38**, **40** has a separate side scrambler **44**, which is generating the same binary sequence. The sequence generated in each channel, however, is offset in time, so that channels are de-correlated locally in time. In a preferred embodiment, decorrelation is achieved over +/−2.9 us, using a pseudo-random binary sequence (PRBS15 sequence), which serves to de-correlate Near End Crosstalk (NEXT) and Far End Crosstalk (FEXT) contributions from other channels. The time offset of the sequences in the respective channels is achieved by loading different seeds in each channel at startup. Separate side scramblers for each channel provide the benefit of keeping global interconnects low while providing a more scalable architecture. Alternatively, a centralized scrambler may be employed.

[0073] The channel logic also includes CRC Encoder logic **46** which is employed to generate two bits from the side scrambler **44** output. The two bits generated by the CRC encoder logic are used by the receive logic to acquire word alignment and inter-channel skew alignment as hereinafter discussed.

[0074] In the present embodiment, a pseudo random binary sequence of length **15** was chosen to allow receiver synchronization within one Idle transmission period. Data in each channel is defined to be constant at either 0x00 or 0xFF during an Idle transmission so the 15 bit seed can be recovered completely within a 16-bit word.

[0075] Two PRBS **15** sequences are employed which are referred to herein as the primary and secondary sequences. The receiver waveform can be de-correlated from the transmitter NEXT if the transmit logic at both ends of the link are using different pseudo-random sequences. The selection of the sequence to be employed at each end of the link may be accomplished in a number of ways. The primary and secondary sequences may be established out of band via a hub. Alternatively, transmission in one direction may be established first using one of the sequences and transmission then established in the opposing direction using the other one of the sequences.

[0076] The two generator polynomials used for the primary and secondary PRBS15 sequences are:

$$G_P(x)=1+x^{14}+x^{15} \quad \text{(Primary generator)}$$

$$G_S(x)=1+x^{11}+x^{15} \quad \text{(Secondary generator)}$$

[0077] These sequences have known linear feedback shift register (LFSR) implementations. The high speed and parallel nature of the PCS **18** employs a 16-bit parallel implementation so as to be able to generate 16 bits per cycle which may be XORed with the incoming data received over the TXP bus. The Side Scrambler **44** is depicted in greater detail in **FIG. 4**.

[0078] The feedback equations describing the primary feedback logic **54** are shown below.

[0079] XPnext0=X1+X2;

[0080] XPnext1=X2+X3;

[0081] XPnext2=X3+X4;

[0082] XPnext3=X4+X5;

[0083] XPnext4=X5+X6;

[0084] XPnext5=X6+X7;

[0085] XPnext6=X7+X8;

[0086] XPnext7=X8+X9;

[0087] XPnext8=X9+X10;

[0088] XPnext9=X10+X11;

[0089] XPnext10=X1I+X12;

[0090] XPnext11=X12+X13;

[0091] XPnext12=X13+X14;

[0092] XPnext13=X0+X1+X14;

[0093] XPnext14=X0+X2;

[0094] XPnext15=X1+X3;

[0095] The "+" signs in the above identified equations indicate exclusive or operations.

[0096] The 15 bit primary seeds to start up a four-channel transmitter up are as follows:

[0097] S_A=0x7FFF (M(0))

[0098] S_B=0x7F00 (M(½))

[0099] S_C=0x70FF (M(¼))

[0100] S_D=0x780F (M(¾))

[0101] To provide the full sixteen bit seed during the initial cycle, the following 16 bit seeds may be employed.

[0102] S_A=0x7FFF (M(0))

[0103] S_B=0x7F00 (M(½))

[0104] S_C=0x70FF (M(½))

[0105] S_D=0x780F (M(¾))

[0106] Where M(x) denotes the relative starting position in the maximum length sequence.

[0107] Similarly, the feedback equations for the secondary sequence are as shown below.

[0108] XSnext0=X1+X5;

[0109] XSnext1=X2+X6;

[0110] XSnext2=X3+X7;

[0111] XSnext3=X4+X8;

[0112] XSnext4=X5+X9;

[0113] XSnext5=X6+X10;

[0114] XSnext6=X7+X11;

[0115] XSnext7=X8+X12;

[0116] XSnext8=X9+X13;

[0117] XSnext9=X10+X14;

[0118] XSnext10=X0+X4+X11;

[0119] XSnext11=X1+X5+X12;

[0120] XSnext12=X2+X6+X13;

[0121] XSnext13=X3+X7+X14;

[0122] XSnext14==X0+X8;

[0123] XSnext15=X1+X9;

[0124] The 15 bit secondary seeds to start up a four-channel transmitter are as follows:

[0125] S_A=0×7FFF (M(0))

[0126] S_B=0×00CC (M(½))

[0127] S_C=0×0AAA (M(¼))

[0128] S_D=0×0008 (M(¾))

[0129] To provide the full sixteen bit seed during the initial cycle, the following seed values may be employed:

[0130] S_A=0×7FFF (M(0))

[0131] S_B=0×00CC (M(½))

[0132] S_C=0×0AAA (M(¼))

[0133] S_D=0×0008 (M(¾))

[0134] Where M(x) denotes the relative starting position in the maximum length sequence.

[0135] It should be appreciated that other suitable starting seeds may be employed for both the primary and the secondary pseudo-random sequence generators.

[0136] The scrambler is described below with respect to the use of the primary seed and primary feedback logic. Operation is the same when utilizing the secondary with the secondary feedback logic noting that the starting seed values differ for the respective channels and the feedback equations differ so that de-correlation is maintained to minimize Far End Crosstalk on the serial data channels.

[0137] The bit seed value for the respective channel is stored within the seed register **42** at startup. In a preferred embodiment, one seed register **42** is provided for each one of the side scramblers and a different seed value (S_A, S_B, S_C, S_D) is stored within each one of the registers. The seed value is passed through a multiplexer **52** and stored within a pseudo-random binary sequence (PRBS) register **52**. The output of the PRBS Register **52** on the bus X<15:0>is exclusive or'd (XORed) with the 16 bit parallel data received over the TXP bus (TXP<15:0) by XOR gates **58** to generate cipher data on the TXC bus (TXC<15:0). It is noted that during the first cycle, if only a fifteen bit seed is employed, the output from the PRBS Register **52** may or may not be valid. Since the primary and secondary logic **54**, **56** generates the next sixteen bit pseudo-random sequence values based upon the low order fifteen bits from the PRBS Register **52**, after the first cycle, the correct sequence value is stored within the PRBS Register. Alternatively, a sixteen bit seed value may be stored within the seed register **42** for each channel so that the correct output within the pseudo-random binary sequence is obtained during the initial cycle as well as all subsequent cycles.

[0138] The output of the PRBS Register **52** is applied to the inputs of the primary feedback logic **52** and the secondary feedback logic **56**. The outputs the primary feedback logic **54** and the secondary feedback logic **56** are coupled to inputs of the multiplexer **50**.

[0139] Assuming again that the primary feedback logic is being employed, the primary feedback logic generates the next 16 bit pseudo-random binary sequence which is clocked into the PRBS Register **52** via the mulitplexer **50**. The new pseudo-random binary sequence stored within the PRBS Register **52** is coupled to the XOR gates **58** via bus X<15:0>and is XORed with the next data word received over the TXP bus (TXP<15:0>) to produce the next 16 bit parallel word of encoded cipher data. This process is repeated for each subsequent word on the TXP bus. As illustrated in **FIG. 3**, The cipher data output from the Scrambler **44** is coupled to CRC encoder logic **46** via the TXC bus. The bits corresponding to TXC<15:0>are passed unchanged to the TX register **48** TXa bus over lines TXa<15:0>in the Channel A logic **34**.

[0140] CRC Block Encoder

[0141] The CRC Block Encoder **46** generates two bits, R**0** and R**1**, from the 16-bit scrambled word received from the scrambler **44**. The R0 bit for Channel A is designated R0a and the R1 bit for Channel A is designated as R1a as illustrated in **FIG. 3**. The bits for the remaining channels (not shown) are designated in a similar manner reflecting their channel association. The two bits are generated using a CRC algorithm. The R0 bit is added directly to the channel data stream as TX15 and is employed by the receive logic to obtain word alignment. The R1 bit is further encoded with control information and is transmitted in a channel other than the one containing the cipher data from which it was derived. Skew alignment and data valid information are decoded from the R1 bits received by the channel reaceive and control logic. More specifically, the R1 bits are encoded with Data Valid information corresponding to the TX_EN<1:0> signals as hereinsfter described. The R0 bit is a function of the 16-bit input word, so the receiver can search for R0 in the recovered stream to obtain correct 18-bit word alignment. The R0 check at the receive logic permits the detection of word misalignment. R0 is calculated so as to assure at least one transition within within each 18-bit word. The worst case run length is 0×00001 followed by 0×30000, which limits the maximum run length to 33.

[0142] The CRC encoder logic **46** processes the received parallel data stream over signal lines TXC<15:0> for each respective channel and generates the R0 and R1 bits. The logic employed by the CRC Block Encoder **46** to generate the R0 and R1 bits for each channel are set forth below.

[0143] R0)=!(TXC0+TXC1+TXC3+TXC4+TXC6+ TXC7+TXC9+TXC10+TXC12+TXC13+TXC15);

[0144] R1=!(TXC0+TXC2+TXC3+TXC5+TXC6+ TXC8+TXC9+TXC11+TXC12+TXC14+TXC15).

[0145] The "+" signs in the above equations indicate exclusive OR functions. The "!" symbol in the above logic equations indicates an inversion. The resultant word and skew alignment bits generated in the above-described manner are inverted both in the transmit path and the receive path in order to assure at least one transition over the serial link for each word transmitted to facilitate clock recovery within the receive logic. These equations can be implemented in 4 levels of 2-input XOR gates, which can operate at high speed

[0146] Skew/Control Encoder

[0147] The R1 bit generated in the above described manner is coupled to skew/control encoder logic **50**. The skew/control encoder logic **50** encodes control information with the R1 bit in each channel and rotates the resultant bit across the channels such that the encoded version of the R1 bit is not transmitted in the same serial channel as the cipher data from which the resultant bit was derived. For example, if R1 is generated in the CRC encoder logic **46** of channel **1**, after encoding with control information, the resultant bit in the illustrated embodiment is transmitted over serial channel **2**.

[0148] Inter-channel skew can occur due to physical path differences as well as varying delay in the deserializer framing. It is therefore necessary to provide a mechanism to correct for such inter-channel skew at the receiver. To minimize overhead, the same bits that are employed to carry information employed to correct inter-channel skew also have encoded thereon control information corresponding to the idle and data valid information received over the TX_EN<1:0> lines. More specifically, there are 4 control states that have to be transmitted across the link (Idle, Data Valid All, Data Valid High, Data Valid Low) to allow byte-wide transmissions. The encoding provided by the skew/control encoder logic produces the TX<17> bit in each channel which is added to the transmit words

[0149] The four control states are received from the SMII bus over the TX_EN<1:0> lines as shown in Table 9. These signals pass through equal pipeline delay logic **35** such that the control information can be reassociated with the respective data. The signals C<1:0> thus represent versions of the TX EN<1:0> signals which are delayed appropriately by delay logic **35**. The signals C<1:0> are coupled to the Skew/Control Encoder logic **50** as depicted in **FIG. 3**.

TABLE 9

C1 (TX_EN1)	C0 (TX_EN0)	Control State
0	0	Idle
0	1	Data Valid Low
1	0	Data Valid High
1	1	Data Valid All

[0150] In the four-channel implementation, the C1 and C0 control bits are encoded with the four R1 bits from each channel ($R_A1 \ldots R_D1$) to produce four control-encoded bits, K<3:0>, as shown below.

[0151] $K3=R_A1+C1$

[0152] $K2=R_B1+C1$

[0153] $K1=R_C1+C0$

[0154] $K0=R_D1+C0$

[0155] Thus, the K3 bit represents the exclusive OR of the R1 bit generated in Channel A with the C1 bit, the K2 bit represents the exclusive OR of the R1 bit generated in Channel B with the C1 bit, the K1 bit represents the exclusive OR of the R1 bit generated in Channel C with the C0 bit and the K0 bit represents the exclusive OR of the R1 bit generated in Channel D with the C0 bit. The code word K<3:0>generated in the above described manner carries the control state information received over signal lines TX_EN<1:0> and allows for the detection of any single-bit error in the R1 bits.

[0156] The Skew/Control Encoder logic **50** rotates the respective K bits to channels in which the R1 bits were not generated to provide a key which the receiver can use to detect and adjust for inter-channel skew. In the present implementation, the K<3:0>code bits are rotated by one bit across the four channels and attached to the data as the TX<17> bit in each channel, as shown in Table 10.

TABLE 10

TX17 Control Encoding

TX17 Control Bits	Encoding
TXa<17>	$K0=R_D1 + C0$
TXb<17>	$K3=R_A1 + C1$
TXc<17>	$K2=R_B1 + C1$
TXd<17>	$K1=R_C1 + C0$

[0157] As illustrated by Table 10 and **FIG. 3**, the R1 bit generated in Channel D is encoded with the C0 bit to form the K0 bit which is forwarded over signal line TXa<17> to the TX Register **48** in Channel A for transmission over Channel A, the R1 bit generated in Channel A is encoded with the C1 bit to form the K3 bit which is forwarded over signal line TXb<17> to the TX Register **48** in Channel B for transmission over Channel B, the R1 bit generated in Channel B is encoded with the C1 bit to form the K2 bit which is forwarded over signal line TXc<17> to the TX Register **48** in Channel C for transmission over Channel C and the R1 bit generated in Channel C is encoded with the C0 bit to form the K1 bit which is forwarded over signal line TXd<17> to the TX Register **48** in Channel D for transmission over Channel D.

[0158] The receiver utilizes the TX<17> bits from the respective channels in combination to to acquire inter-channel sync. Additionally, the receiver uses the TX<17> bits received over the respective channels to extract the control information during normal operation as hereinafter described.

[0159] The TX Register **48** for Channel A drives the signal lines TXa<17:0>. The signal lines TXa<17:0> are derived from the output of the CRC encoder TXa<15:0>. The R0a bit is coupled through the TX Register **48** to form TXa<16> and the skew/control encoder logic **50** generates the K0 signal which is coupled to the bus TXa<17:0> via the TX Register **48**.

[0160] The signal lines TX<17:0> at the output of the TX Register **48** for each of the four channels are coupled to the input to the Serializer/Deserializer logic **24***a*-**26***d*. A differential transmit clock TX_CLK+, TL_CLK−, also driven by the physical coding sublayer **18**, is also included within the SERDES bus **22** for each channel. A serializer within the SERDES logic for each channel receives as an input the 18 bit parallel encoded data and serializes the data to form a serialized bit stream. The serialized bit stream is transmitted from the respective serializers (**24***a*-**24***d*) over the respective communication links or channels **26***a*-**26***d*. In the illustrated embodiment, the serial links comprise differential signals to the driver logic **28**.

[0161] The serialized data are transmitted over the serial link over signal lines TXD+/− and are received at inputs RXD+/− from receive logic **28** which adapts the serial signals to the applicable transmission medium. The receive data is converted by the deserializer from serial form to an 18 bit parallel data word within deserializer logic **24** and transmitted over the receive portion of the SERDES bus **22** to the physical coding sublayer **18**. The parallel data is then processed within the receive path of the physical coding sublayer as discussed below to obtain word and skew alignment.

[0162] Receive Path

[0163] The receive path through the physical coding sublayer **18** is depicted with greater particularity in **FIG. 5**. The receive path roughly follows the transmit path in reverse with the added complexity needed to obtain word alignment within each channel and skew alignment among the channels, and to obtain seed synchronization for the descrambler.

[0164] Word and skew alignment is performed in a number of steps. As the initial step in acquiring synchronization of received data, word-frame alignment within each channel is achieved using the R0 bits. The channels are then de-skewed using the received K bits and calculated R1 bits. The scrambler states are then acquired during an Idle transmission. The Bit Error Rate during acquisition is assumed to be reasonably good such that there is a high probability that correlation calculations done over 128 words will not experience an error.

[0165] Although the receive channel logic **80** is depicted and described specifically for Channel A, is should be appreciated that similar receive channel logic, though not shown for ease of illustration, is provided for each of the other receive channels.

[0166] Receiver Word Frame Alignment

[0167] The apparatus and techniques for performing word framing are illustrated in **FIGS. 5 and 6**. Word framing within each channel, is acquired by checking the calculated or estimated R0 bit out of the CRC Decoder **90** against the transmitted R0 bit (RX<16>). The Word Frame Control machine **84** in each channel shifts the framing until acquisition in each channel is achieved. The R0 bit is derived from the scrambled data which can be considered to have a random distribution. The Word Frame Control correlates (XORs) the transmitted value against the calculated value for a predetermined number of samples to declare alignment with a low probability of false detection. More specifically, alignment may be declared if a predetermined number of matches are detected over a predetermined number of word samples.

$$\sum_{n=1}^{N} \hat{R}0_n + R0_n = 0$$

[0168] Where $\hat{R}0$ is the estimated value of R0 generated from the receiver CRC Decoder. The probability of false alignment is 3E-39 if the calculation is done over 128 words (N=128) and the scrambled data is random.

[0169] The received data for each channel when presented to the physical coding sublayer **18** is framed randomly by the de-serializer logic **24**. One technique for performing word framing is depicted with greater particularity in **FIG. 6**. The received word which is delayed in a register **120** to produce a delayed word and fed into a multiplexer **124** along with the current received word held in register **122**. The multiplexer thus has access to 36 contiguous bits of the serial stream and can re-frame across the arbitrary word boundary presented at the physical coding sublayer **18** input. The word frame control machine **84** uses the CRC Decoder **90** outputs to search through the 18 possible framing options until the estimated correct frame location is acquired. This framing is thus verified over a predetermined number of word samples.

[0170] Receiver Inter-Channel Alignment

[0171] Once word framing is achieved in all channels the bits are assumed to be correct into the Control Decoder **92** and Inter-Channel alignment can proceed. There are several methods of proceeding but the approach outlined below aligns channels A, B and C before aligning D. In a similar process to that used in word framing, correlation of the bits to the data can be checked and any control data discarded.

$$\sum_{n=1}^{N} \hat{R}1a_n + \hat{R}1b_n + K2_n + K3_n = 0$$

$$\sum_{n=1}^{N} \hat{R}1a_n + \hat{R}1b_n + R1b_n + C1_n + R1a_n + C1_n = 0$$

$$\sum_{n=1}^{N} \left(\hat{R}1a_n + R1a_n\right) + \left(\hat{R}1b_n + R1b_n\right) = 0$$

[0172] Where $\hat{R}1a$ and $\hat{R}1b$ are the estimated or calculated values of R1a and R1b generated from the CRC Decoders in channels A and B respectively. K2 and K3 are the encoded

bit generated by the skew/control encoder **50** and transmitted in channels C and B respectively.

[**0173**] Inter-channel skew between A, B and C can be adjusted until the correct correlation occurs. Inter-channel skew among channels A, B and C is adjusted until the correct correlation occurs. Inter-Channel skew for the three channels is verified over a number of samples to verify to a high probability that the words in the three Channels are properly aligned. In the event the Inter-Word align control state machine **88** determines that the words in the three channels are not aligned, word framing on one of the channels is adjusted backward or forward by one word in successive steps according to a prespecified method until proper word alignment across the channels is obtained. The interchannel word alignment may be adjusted by inter-word Align Control Logic **88** in any suitable manner until proper alignment across the three channels is obtained. One exemplary approach in Table 11 below.

Channel	Step	0	1	2	3	4	5	6	7	8
A		0	0	0	0	0	0	0	0	0
B		0	−1	+1	0	0	−1	−1	+1	+1
C		0	0	0	−1	+1	−1	+1	−1	+1

Table 11

[**0174**] While the above table illustrates steps within channel B and C for −1 word and +1 word skew adjustments, the table may be extended to −2 and +2 word skew adjustments or as far as necessary to assure that any possible skew adjustments can be accommodated within a given system.

[**0175**] After obtaining alignment of channels A, B and C in the above described manner, a similar calculation can be performed by the inter word align controller **88** to assure that channel D is properly aligned with channels A, B and C. The logic necessary to perform such verification is illustrated by the following logic equations.

$$\sum_{n=1}^{N} \hat{R}1c_n + \hat{R}1d_n + K0_n + K1_n = 0$$

$$\sum_{n=1}^{N} \hat{R}1c_n + \hat{R}1d_n + R1c_n + C0_n + R1d_n + C0_n = 0$$

$$\sum_{n=1}^{N} \left(\hat{R}1c_n + R1c_n\right) + \left(\hat{R}1d_n + R1d_n\right) = 0$$

[**0176**] In particular, the calculated R1 bit for Channel C is exclusive ORed with the calculated R1 bit for channel D, the K0 bit transmitted over Channel A and the K1 bit transmitted over Channel D over N words. As illustrated by the logic equations above, when channel D is properly aligned with Channels A, B, and D the value of the above identified exclusive OR operation will be 0. By repeating the verification over a predetermined number of sample words, alignment of channels A, C and D can be assured to an extremely high probability.

[**0177**] In the event the test of Channel D alignment reveals that it is not aligned with the other two channels included in the exclusive OR operation, the word alignment of Channel D is adjusted and the verification test is repeated until alignment is achieved. For example, if the initial verification D for Channel D alignment reveals that it is not aligned, the word alignment on Channel D may be adjusted backwards by one word (−1 word). If the verification test for alignment on Channel D again reveals no alignment, the alignment of Channel D may be adjusted forward 1 word from its initial framing alignment and the verification test repeated. This adjustment process may be repeated until proper skew alignment for channel D is achieved.

[**0178**] **FIG. 7** shows an exemplary implementation to achieve interchannel skew alignment. Each channel contains a cascade of three delay registers **130**, **132**, **134** which progressively delay the receive words. The number of delay stages required is a function of the possible skew that could occur during transmission through the channel and is system specific. The implementation in **FIG. 7** allows for an interchannel skew of +/−1 word. A multiplexer **136** in each channel can select the appropriate delayed word as the output and the Inter-Channel Alignment Control **88** adjusts the delays until the K bits are received correctly according to the equations above.

[**0179**] Receiver Scrambler Acquisition

[**0180**] Once word and skew alignment is achieved, the receiver can detect when an Idle symbol is being transmitted. The Idle state is defined to be 0x5555555555555555 on the SMII which appears as 0xFF in channels A and C and 0x00 in channels B and D after interleaving. The Idle is then scrambled in each channel. The scrambler seed can be recovered in each channel by XORing the received cipher text, RXC<15:0>, with either 0x00 or 0xFF in the appropriate channel. The pseudo random sequence state can be completely recovered in one Idle period because the sequence length is 15 and the complete state is defined in each channel's 16-bit data word.

[**0181**] Receiver Error Detection

[**0182**] Once the receiver has acquired complete lock (as described in the above described acquisition process) with a high degree of certainty, the receiver can recover the data and control and use the R0 and K bits for error detection. State machines can keep track of the error rate in the receiver and make a determination of when re-acquisition is required. The high data rates and continuous monitoring of alignment on each symbol make error detection and re-acquisition very fast and blind to the transmitter.

[**0183**] The Receive Interleaver

[**0184**] The outputs from the four channels RXPa<15:0>, RXPb<15:0>, RXPc<15:0>, RXPd<15:0> are coupled to the receive interleave logic **106** which recombines the data to service the SMII receive bus. The data from the four channels are combined by the receive interleave logic **106** as indicated below to form the original 64 bit word.

[**0185**] RXD<63:0>=<RXPd15, RXPc15, RXPb15, RXPa15, RXPd14, RXPc14, RXPb14, RXPa14, RXPd13, RXPc13, RXPb13, RXPa13, RXPd12, RXPc12, RXPb12, RXPa12, RXPd11, RXPc11, RXPb11, RXPa11, RXPd10, RXPc10, RXPb10, RXPa10, RXPd9, RXPc9, RXPb9, RXPa9, RXPd8, RXPc8, RXPb8, RXPa8, RXPd7, RXPc7, RXPb7,

RXPa7, RXPd6, RXPc6, RXPb6, RXPa6, RXPd5, RXPc5, RXPb5, RXPa5, RXPd4, RXPc4, RXPb4, RXPa4, RXPd3, RXPc3, RXPb3, RXPa3, RXPd2, RXPc2, RXPb2, RXPa2, RXPd1, RXPc1, RXPb1, RXPa1, RXPd0, RXPc0, RXPb0, RXPa0>

[0186] The 64 bit data word, the control signals originally received over the TX_EN<1:0> lines and an indication of an error condition, if any has been detected in the course of transmission over the physical coding sublayer, is forwarded by the interleave logic **106** to the receive control logic **108**.

[0187] The above described physical coding sublayer **18** may be fabricated as a single integrated circuit to minimize the circuit board real estate dedicated to the presently described data transmission and distribution functions. Moreover, the SERDES logic may also be integrated with the physical coding sublayer logic so that all common logic normally associated with the physical coding sublayer system is integrated in a single device. Given such integration, off chip logic would needed to interface the device to the specific media employed for the multiple serial channels and logic necessary to interface the wide parallel interface to reconciliation logic necessary to interface the device to MAC circuitry or any other desired system interface.

[0188] It should be noted that the scrambler logic may be omitted in the event decorrelation of the transmitted data is not required. Additionally, it should be noted that in another embodiment the control values represented by the C bits need not be encoded on the R1 bits. In such event the skew alignment bit transmitted over the serial channel comprises the R1 bits rather than control encoded versions of such bits. The transmission of the R1 bits as the skew alignment bits simplifies the skew alignment process and allows alignment to proceed two channels at a time rather than three at a time as described above. Additionally, CRC encoding and decoding logic may be reduced by generating a single CRC code for using the single code for generation of both the word framing and skew alignment.

[0189] Finally, it will be understood by those of ordinary skill in the art that modifications to and variations of the above described methods and apparatus may be made without departing from the inventive concepts disclosed herein. Accordingly, the invention should not be viewed as limited except as by the scope and spirit of the appended claims.

What is claimed is:

1. A method for transporting data across a plurality of data channels comprising:

concurrently generating a plurality of lesser width parallel data words containing parallel data from a greater width parallel data word such that all adjacent bits of said greater width parallel data word are contained in different ones of said lesser width parallel data words, wherein the number of bits in said greater width parallel data word is greater than the number of bits in each of said lesser width parallel data words;

serializing parallel data representative of said plurality of lesser width parallel data words; and

transmitting said serialized data words over a corresponding plurality of distinct serial data channels.

2. A method for transporting data across a plurality of data channels comprising:

concurrently generating a plurality of lesser width width parallel data words containing parallel data from a greater width parallel data word such that all adjacent bits of said greater width parallel data word are contained in different ones of said lesser width parallel data words, wherein the number of bits in said greater width parallel data word is greater than the number of bits in each of said lesser width parallel data words;

scrambling the parallel data in said lesser width parallel data words to form a plurality of scrambled data words;

serializing said scrambled data words; and

transmitting said serialized scrambled data words over a corresponding plurality of distinct serial data channels.

* * * * *