Report Number: WUCS-90-14

1990-03-01

Testing the Taxi

William D. Richard

The Advanced Micro Devices TAXI chip set was chosen as the serial interface for the Fast Packet Switch. In order to fully understand the operation of the chip set, and in order to fully characterize the chip set's operation, several tests were performed using actual TAXI components. These tests included TAXI-to-TAXI tests and tests with optical data links. The operation of the asynchronous transmitter handshake was also investigated in detail. The results of these tests indicate that the TAXI chip set is well built and very reliable.

TESTING THE TAXI

William D. Richard

WUCS-90-14

March 1990

Department of Computer Science
Washington University
Campus Box 1045
One Brookings Drive
Saint Louis, MO 63130-4899

Dr. Richard's work was supported by Southwestern Bell Telephone.

Abstract

The Advanced Micro Devices TAXI chip set was chosen as the serial interface for the Fast Packet Switch. In order to fully understand the operation of this chip set, and in order to fully characterize the chip set's operation, several tests were performed using actual TAXI components. These tests included TAXI-to-TAXI tests and tests with optical data links. The operation of the asynchronous transmitter handshake was also investigated in detail. The results of these tests indicate that the TAXI chip set is well built and very reliable.

1 Introduction

The Advanced Micro Devices TAXI (Transparent Asynchronous Xmitter-Receiver Interface) chip set provides a means to establish a transparent, high-speed serial link between two high-performance parallel buses [1]. The TAXI chip set consists of a Transmitter, shown in Fig. 1, which takes parallel data and transmits it serially at up to 125 MHz, and a receiver, shown in Fig. 2, which converts the serial data stream back into parallel form.

The TAXI transmitter consists of an input latch, an encoder, a parallel-to-serial shift register, a multiplying Phase Locked Loop (PLL), and some control logic. Data is input to the latch, encoded, and shifted out at the serial data rate. The transmitter uses the 4B/5B scheme associated with the ANSI X3T9.5 Fiber Distributed Data Interface (FDDI) specification. This encoding divides an 8-bit byte into two 4-bit nibbles. Each nibble is encoded into a 5-bit symbol. The new 10-bit encoded "byte" is then further formatted into a Non-return to zero, invert on ones (NRZI) data stream for the output media. This 4B/5B encoding is 80% efficient, i.e., it uses a 125 MHz transmission rate to send 100 M-bits of data per second.

The Am7968 transmitter has differential pseudo-ECL (referenced to +5V) outputs which can drive 50 Ohm terminated lines. This capability makes it easy to interface directly to shielded twisted pair or coaxial cables. The pseudo-ECL outputs are also compatible with the ECL interfaces of popular optical data links (ODLs) used to drive multi-mode fiber optic cable. The NEC NEOLINK-1312 ODL [2] and the AT&T ODL 200 series transmitter/receiver pair [3] interface directly to the TAXI chip set using these pseudo-ECL signal levels.

The TAXI receiver accepts the encoded data stream into a serial-to-parallel converter, decodes the data stream, and outputs the received data with an accompanying strobe. An on-chip data tracking PLL performs the required clock recovery from the input serial data stream. The receiver, like the transmitter, uses pseudo-ECL signal levels on its serial data inputs.

The TAXI chip set was chosen as the serial interface for the Fast Packet Switch. In order to fully understand the operation of TAXI chip set, and in order to fully characterize the chip set's operation, several tests were performed using actual TAXI components. This technical report describes the results of that testing.

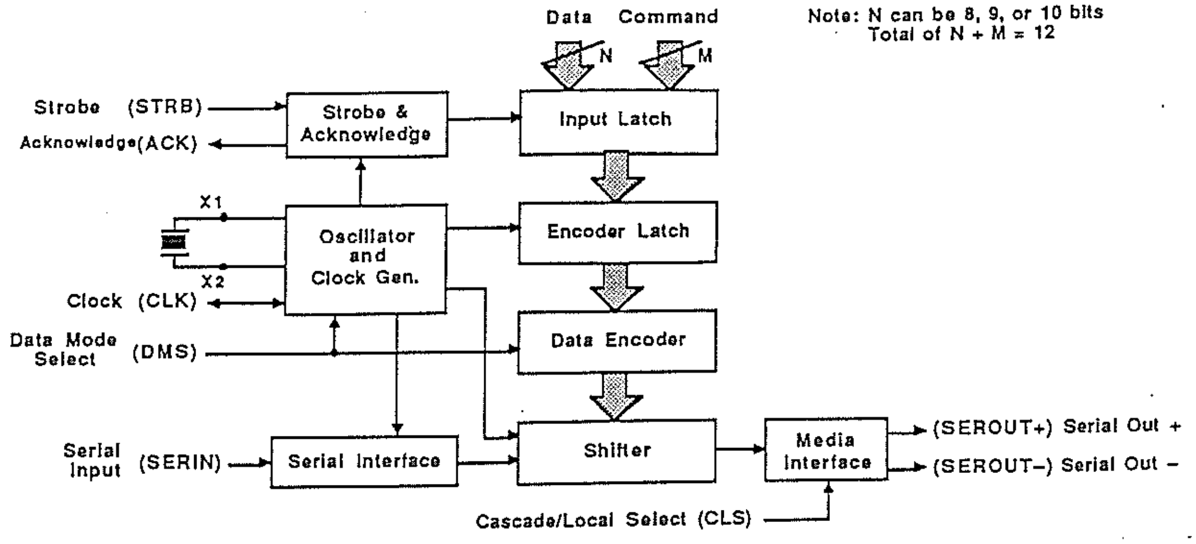

Fig. 1. The Am7968 TAXI Transmitter

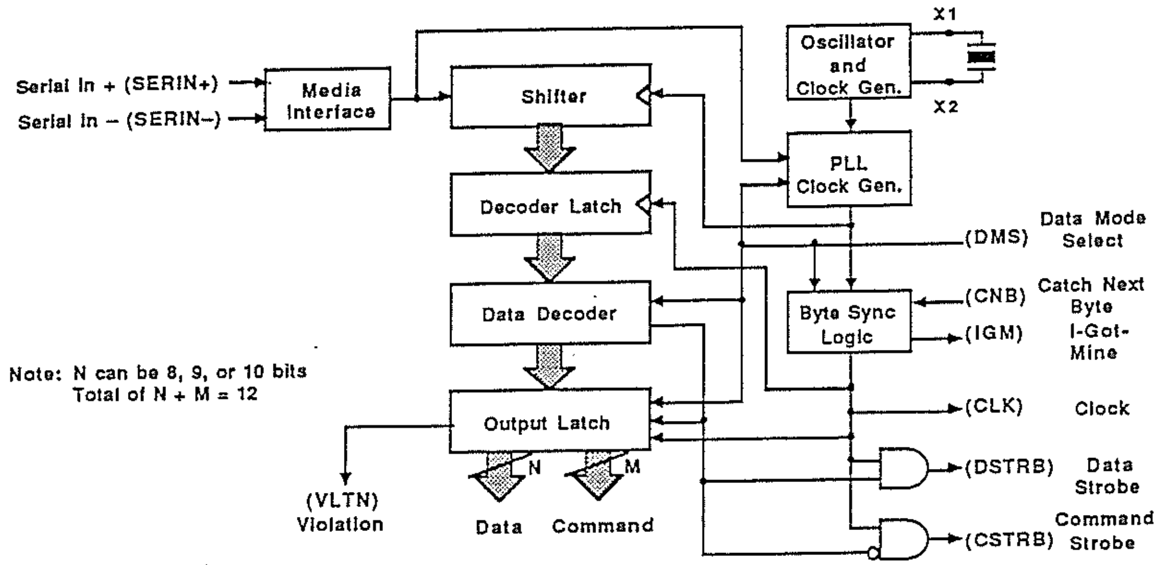

Fig. 2. The Am7968 TAXI Receiver

2 The TAXI Digital Link Test System

In order to provide an appropriate test bed for testing the performance of a TAXI digital link, a TAXI digital link test system was designed and implemented. This system consists of an Advanced Micro Devices (AMD) Am7968 TAXI transmitter subsystem, an AMD Am7969 TAXI receiver subsystem, and an IBM PC-based interface. The overall architecture of the TAXI digital link test system is shown in Fig. 3.

The TAXI transmitter subsystem consists of a 35 ns, 2 K-byte, 8-bit, parallel FIFO, an Am7968 TAXI transmitter, a 12.5 MHz crystal clock oscillator circuit, and an asynchronous controller as shown in Fig. 4. The FIFO accepts 8-bit input data from the IBM PC interface at normal PC bus speeds. It outputs 8-bit data, under control of the asynchronous controller, to the TAXI transmitter at a 12.5 M-byte per second rate. Stop/start control for the asynchronous controller is provided by the PC interface via a two-stage synchronizer clocked by the transmitter clock. This allows the controlling software program running on the PC to disable transmission, fill the FIFO, and then restart transmission. In this manner, it is possible to transmit 2 k-byte data packets across the TAXI link at the full 12.5 M-byte per second rate.

The TAXI receiver subsystem consists of a 35 ns, 2 K-byte, 8-bit, parallel FIFO, an Am7969 TAXI receiver, a 12.5 Mhz crystal clock oscillator circuit, and a controller. Data received by the TAXI receiver from the digital link is strobed into the FIFO by the inverted TAXI DSTRB (data strobe) signal, i.e., a single gate forms the controller for this end of the digital link. Data is read from the FIFO via the PC interface. A PC input port is used to read the status of a special flip-flop that detects transitions on the TAXI receiver's VLTIN (violation) output pin. Any pattern which does not decode as a valid command or data pattern is flagged by the receiver using this pin.

The IBM PC-based system provides the interface between the control/test software and the digital link. An output port provides write capability to the FIFO in the TAXI transmitter subsystem, and an input port provides read capability from the FIFO in the TAXI receiver subsystem. A second output port is used to generate system-wide control signals, i.e., reset, start/stop transmission, etc. A second read port is used to read the status of the VLTIN flop-flop in the receiver subsystem. A 0-to-1 transition on the violation pin of the receiver sets this flip-flop. The flip-flop can be cleared by a reset signal from the output control port. The second input port is also used to read the status of the FIFO's empty flag (EF-/DATA AVAIL). A logic '1' on this pin indicates that data is available and can be read by the PC.

During initial testing, the serial outputs of the TAXI transmitter were connected, with proper termination, to the serial inputs of the TAXI receiver. This configuration allowed direct testing of TAXI performance without influence from ODLs. During the second phase of testing, ODLs were installed and the performance of a TAXI-to-ODL-to-ODL-to-TAXI link measured.

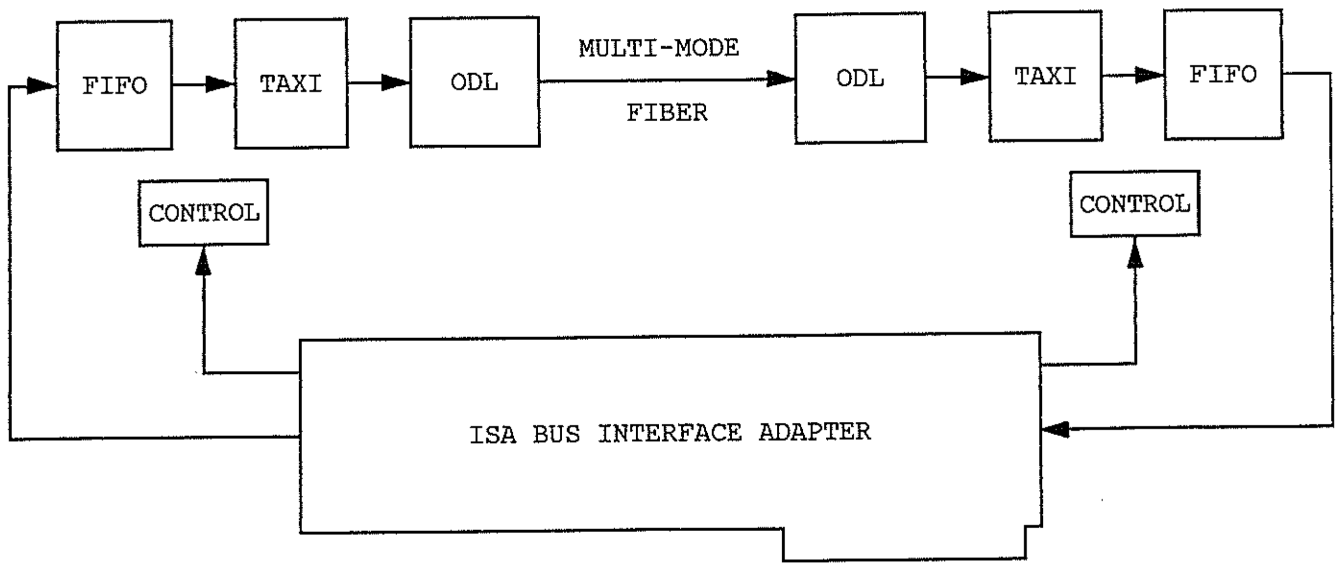

Fig. 3. The Digital Link Test System

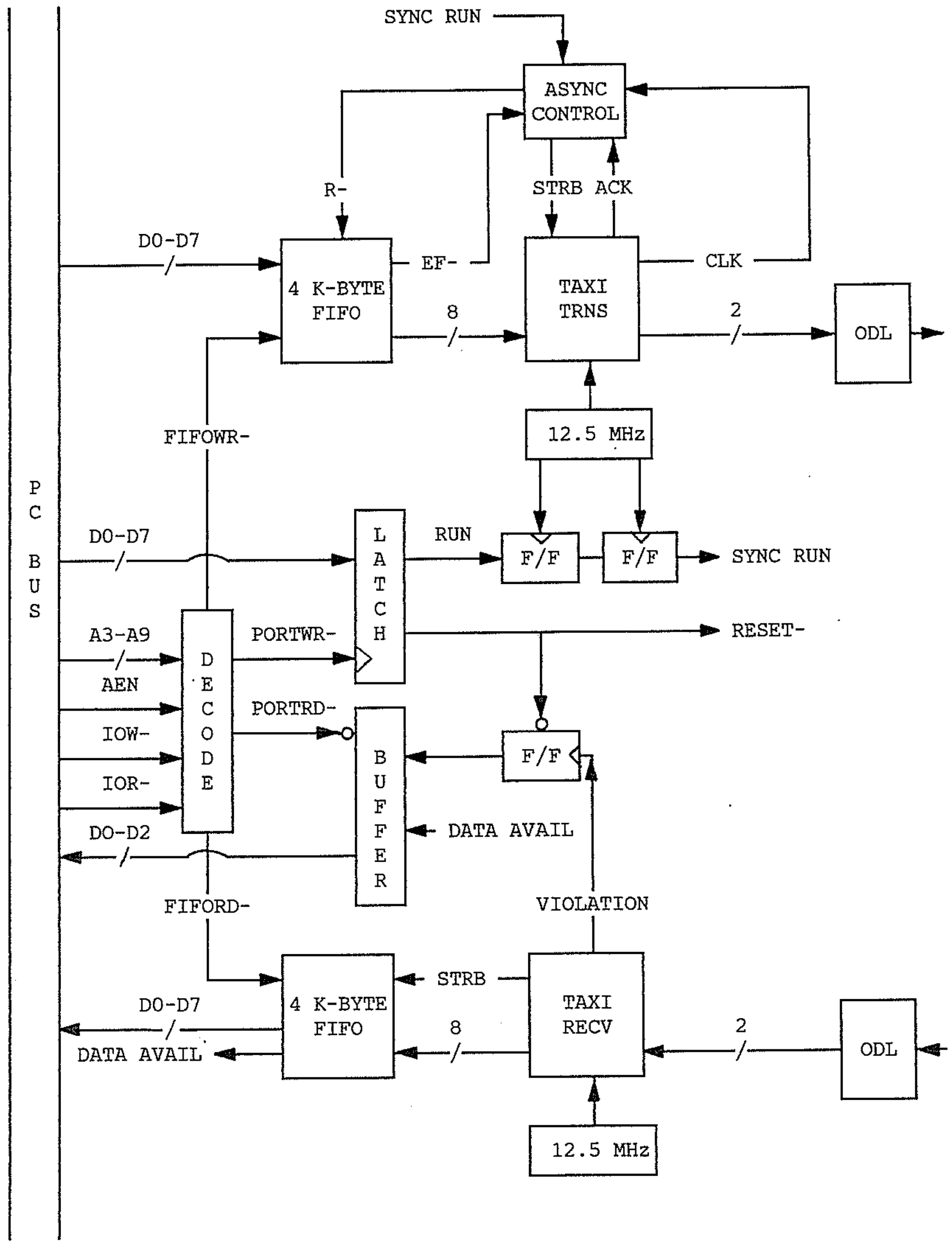

Fig 4. Detail of the Digital Link Test System

The controlling software for the digital link was originally written in Turbo Pascal [4]. This software, which is shown in Fig. 5, sent random 2 K-byte packets across the link and checked the received data for errors. Each packet transmission was allowed to complete before the verification process was started. This eliminated possible metastability problems in the FIFOs from affecting the test results. In order to improve performance, the original routines were later recoded in Turbo C [5] and speed critical portions coded in assembly language. The combination of Turbo C and assembly language allowed 1000 2 K-byte packets to be passed through the link and verified for correctness every 50 seconds using an 8 MHz PC/XT clone.

3 TAXI-to-TAXI Link Tests

The digital link test system has been used extensively to test the TAXI-to-TAXI digital link. Initial testing involved transmission of 2 k-byte packets of random data using the software shown in Fig. 5. Packets were generated using a software random number generator and loaded into the TAXI transmission subsystem FIFO. The asynchronous controller was then enabled, via the synchronizer, and the packet sent across the differential ECL link to the TAXI receiver, which wrote each byte into the output FIFO. The contents of the output FIFO were then read by the test software and compared with the transmitted data. In addition, the status of the violation detection flip-flop was checked for each packet transmitted.

In order to speed up the test process, each random packet was transmitted and verified 1000 times before a new packet was generated. This drastically reduced the overhead associated with random packet generation.

In all, 5 million 2 k-byte packets, or 81.92 billion bits, were transmitted and verified in the manner described above. No transmission errors or violations were detected during these tests. This indicates that the bit error rate for a TAXI-to-TAXI link is less than 1 in 10 billion.

A function generator was next used to check the range of input frequencies acceptable by the TAXI receiver given a fixed 12.5 MHz transmitter clock. The receiver performed without error during initial tests with clock frequencies in the range from 12.1 MHz to 12.9 MHz. Above 12.9 MHz, or below 12.1 MHz, the TAXI receiver failed continuously. Since the TAXI receiver is not designed for this type of use, these tests were not as extensive as the other tests.

Extensive tests were performed to verify the operation of the asynchronous data handshake used by the TAXI transmitter. Tests with an adjustable delay line identified a 200 ps region, relative to the buffered TAXI transmitter output clock, where the TAXI transmitter decides whether or not to accept, via an asynchronous handshake, a byte during the current clock cycle. In order to verify that failure did not occur when the TAXI was clocked during this 200 ps window, due, for instance, to metastability, a test was devised using the adjustable delay line that continually placed the data strobe within the 200 ps window.

```
program Taxi ; { Test the TAXI digital link using random packets.          }

var I, J, K,                               {Loop counters...                  }
    AValue,                                {Temporary variable...             }
    BadPackets,                            {Total bad packets so far...       }
    BadBytes,                              {Total bad bytes so far...         }
    BadBits,                               {Total bad bits so far...          }
    PacketByteErrors,                      {Total byte errors in a packet...}
    PacketBitErrors,                       {Total bit errors in a packet... }
    Violations : Integer ;                 {Total violations so far...        }
    DataArray :  array[0..2045] of byte ;  {The random data array...          }
    Flag : boolean ;                       {Hardware reset flag...            }

begin

   Flag := false ;                         {Set reset flag to false...        }
   BadPackets := 0 ;                       {Set number of bad packets to 0    }
   BadBytes := 0 ;                         {Set number of bad bytes to 0      }
   BadBits := 0 ;                          {Set number of bad bits to 0       }
   Violations := 0 ;                       {Set number of violations to 0     }
   port[$309] := 0 ;                       {Set to initial state...           }
   port[$309] := 4 ;                       {Set Reset high...                 }
   port[$309] := 6 ;                       {Set R- low...                     }
   port[$308] := $a5 ;                     {Send initial garbage byte...      }

   for K := 1 to 30000 do                  {Use 30000 random packets...       }
   begin

      for I := 0 to 2045 do                {Initialize random data...         }
      begin
         DataArray[I] := trunc(Random * 255) ;
      end ;

      for J := 1 to 1000 do                {Each packet sent 1000 times...    }
      begin

         PacketByteErrors := 0 ;           {Set number of errors to 0...      }
         PacketBitErrors := 0 ;

         for I := 0 to 2045 do             {Write 2K - 2 bytes to FIFO...     }
         begin
            port[$308] := DataArray[I] ;
         end ;

         port[$308] := $a5 ;               {Write the req. garbage byte...    }
         port[$309] := 7 ;                 {Set Run high...                   }
         Delay(1) ;                        {Delay 1 millisecond...            }
         AValue := Port[$309] and 2 ;      {Check for violation...            }
         if AValue = 2 then
         begin
            Violations := Violations + 1 ;
            Writeln('Packet ',J:7,', Loop ',K:7,' had a violation.') ;
            Flag := true ;
         end ;
```

Fig. 5. TAXI Digital Link Test Program - Pascal Version

```
AValue := Port[$309] and 1 ;       {Check for data available...      }
if AValue = 1 then
begin
   AValue := Port[$308] ;          {Read and check garbage byte...   }
   if (AValue <> $a5) then
   begin
      PacketByteErrors := PacketByteErrors + 1 ;
      AValue := AValue xor $a5 ;
      If (AValue and $01) = $01 then
         PacketBitErrors := PacketBitErrors + 1 ;
      If (AValue and $02) = $02 then
         PacketBitErrors := PacketBitErrors + 1 ;
      If (AValue and $04) = $04 then
         PacketBitErrors := PacketBitErrors + 1 ;
      If (AValue and $08) = $08 then
         PacketBitErrors := PacketBitErrors + 1 ;
      If (AValue and $10) = $10 then
         PacketBitErrors := PacketBitErrors + 1 ;
      If (AValue and $20) = $20 then
         PacketBitErrors := PacketBitErrors + 1 ;
      If (AValue and $40) = $40 then
         PacketBitErrors := PacketBitErrors + 1 ;
      If (AValue and $80) = $80 then
         PacketBitErrors := PacketBitErrors + 1 ;
   end ;
end
else
begin
   PacketByteErrors := PacketByteErrors + 1 ;
   PacketBitErrors := PacketBitErrors + 8 ;
end ;

for I := 0 to 2045 do              {Read and check data...           }
begin
   AValue := Port[$309] and 1 ;    {Check for data available...      }
   if AValue = 1 then
   begin
      AValue := Port[$308] ;
      if (AValue <> DataArray[I]) then
      begin
         PacketByteErrors := PacketByteErrors + 1 ;
         AValue := AValue xor DataArray[I] ;
         If (AValue and $01) = $01 then
            PacketBitErrors := PacketBitErrors + 1 ;
         If (AValue and $02) = $02 then
            PacketBitErrors := PacketBitErrors + 1 ;
         If (AValue and $04) = $04 then
            PacketBitErrors := PacketBitErrors + 1 ;
         If (AValue and $08) = $08 then
            PacketBitErrors := PacketBitErrors + 1 ;
         If (AValue and $10) = $10 then
            PacketBitErrors := PacketBitErrors + 1 ;
         If (AValue and $20) = $20 then
            PacketBitErrors := PacketBitErrors + 1 ;
```

Fig. 5. (Continued)

```
               If (AValue and $40) = $40 then
                  PacketBitErrors := PacketBitErrors + 1 ;
               If (AValue and $80) = $80 then
                  PacketBitErrors := PacketBitErrors + 1 ;
            end ;
         end
         else
         begin
            PacketByteErrors := PacketByteErrors + 1 ;
            PacketBitErrors := PacketBitErrors + 8 ;
         end ;
      end ;

      if (PacketByteErrors <> 0) then
      begin
         BadPackets := BadPackets + 1 ;
         BadBytes := BadBytes + PacketByteErrors ;
         BadBits := BadBits + PacketBitErrors ;
         Writeln('Packet ',J:7,', Loop ',K:7,' had ',PacketByteErrors:7,
                 ' bad bytes, ',PacketBitErrors:7,' bad bits') ;
         Flag := true ;
      end ;

      if (Flag) then
      begin
      port[$309] := 0 ;                  {Set to initial state...          }
      port[$309] := 4 ;                  {Set Reset high...                }
      port[$309] := 6 ;                  {Set R- low...                    }
      port[$308] := $a5 ;                {Send initial garbage byte...     }
      Flag := false ;
      end
      else
      begin
         port[$309] := 6 ;               {Set Run low...                }
      end ;

   end ;

   Writeln('After  ',K:7,' loops, bad packets = ',BadPackets:7,
           ', violations = ',Violations:7) ;
   Writeln('                      bad bytes =   ',BadBytes:7,
           ', bad bits =   ',BadBits:7) ;
 end ;

end .
```

Fig. 5. (Continued)

The modified digital link controller was used to transmit 1 million 2 k-byte packets, or 16.384 billion bits, across the link at each of five strobe positions relative to the start of the 200 ps window. These strobe positions were spaced at equal increments across the window, 50 ps apart. No errors were detected during this test.

The spacing of test points was then decreased to 20 ps and 20 million bytes, or 160 million bits, transmitted across the link for each strobe position. This additional testing was performed using one-byte packets to ensure that a decision was actually being made for each byte transmitted. Again, no errors were detected.

To insure the validity of the digital link test results, circuit and software verification was performed. Extensive testing was performed on the link to verify its correct operation. This included introducing errors in the transmitted data using an oscilloscope probe to insert noise into the differential ECL signals, for instance, and verifying that the error was correctly identified by the system. Errors were also detected during the TAXI receiver frequency range test discussed above when the receiver frequency was varied beyond the lock limits of the internal phase-locked loop.

4 Using ODLs with the TAXI Chip Set

The Taxi digital link test system was modified in order to provide an appropriate test bed for testing the performance of commercially available multi-mode ODLs. Interfaces were designed and implemented for the NEC NEOLINK-1312 transmitter/receiver and the AT&T ODL 200 1252N transmitter and 1352N receiver pair.

The circuitry necessary to interface the NEC NEOLINK-1312 ODL to the TAXI transmitter/receiver pair on the digital link test system was developed and implemented initially. The pin connections for the NEOLINK-1312 are shown in Fig. 6 [2]. Both the TAXI chip set and the NEC ODL use ECL signal levels referenced to +5 Volts. This allowed the SEROUT+ and SEROUT- signals from the TAXI transmitter to be connected directly to the DATA and DATA- inputs on the NEC ODL using a twisted pair connection. In each case, a 475 Ohm pull-down resistor was used between the signal line and ground. This type of connection scheme was acceptable since the TAXI transmitter and the ODL were separated by less than three inches. Similarly, the DATA and DATA- outputs from the NEC ODL were connected directly to the SERIN+ and SERIN- inputs on the TAXI receiver using a twisted pair connection. A single 475 Ohm pull-down was used between each signal line and ground. Pull-down resistors were also used on the SIGDET and SIGDET- (signal detect) outputs of the NEC ODL in order to create appropriate signal levels for testing purposes.

Circuitry was also developed to interface the AT&T ODL 200 transmitter and receiver pair to the TAXI transmitter/receiver pair on the digital link test system. The block diagrams of the AT&T ODL pair are shown in Fig. 7. Again, both the TAXI chip set and the AT&T ODLs use ECL signal levels referenced to +5 Volts. This allowed the SEROUT+ and SEROUT- signals from the TAXI transmitter to be connected directly to the DATA and DATA- inputs on the

Pin No.	Receiver block	Pin No.	Transmitter block
1~3	Rx V_{EE} (Case)	11~14	Tx V_{EE} (Case)
4~6	Rx V_{CC}	15	Tx $\overline{\text{DATA}}$ input
7	Rx DATA output	16	Tx DATA input
8	Rx $\overline{\text{DATA}}$ output	17~18	Tx V_{EE} (Case)
9	Rx $\overline{\text{Signal Detect}}$ output	19~20	Tx V_{CC}
10	Rx Signal Detect output		

Fig. 6. Pin Connections for the NEC NEOLINK-1312 ODL

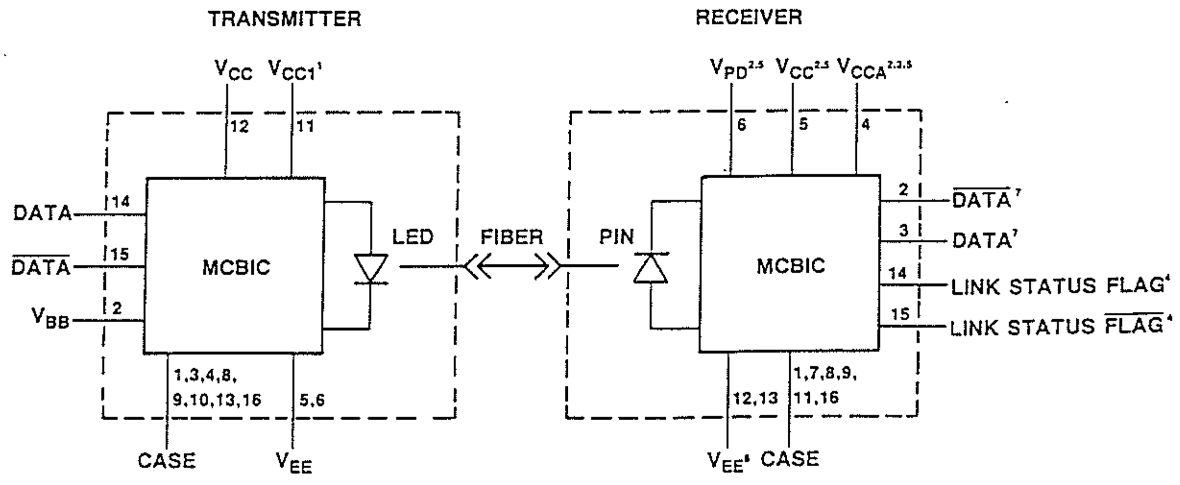

Fig. 7. Block Diagram of the AT&T ODL 200 Transmitter and Receiver

1252N transmitter. In each case, a 475 Ohm pull-down resistor was used between the signal line and ground. Similarly, the DATA and DATA- outputs from the 1352N receiver were connected directly to the SERIN+ and SERIN- inputs on the TAXI receiver. A single 475 Ohm pull-down was used between each signal line and ground. Pull-down resistors were also used on the FLAG and FLAG- outputs of the 1352N receiver in order to create appropriate signal levels for testing purposes. These outputs indicate the presence or absence of a minimum acceptable level of optical signal input.

Currently, circuitry is being developed to interface the AT&T 1227 single-mode transmitter and the AT&T 1310 single-mode receiver to the TAXI digital link test system. Once the TAXI digital link test system is operational using these single-mode devices, the fiber links between the Washington University School of Engineering and Applied Science and the Washington University Electronic Radiology Laboratory will be tested using a loop-back connection.

5 Testing ODL Performance

The TAXI digital link test system was used extensively to test the TAXI-to-ODL-to-ODL-to-TAXI digital links described above. A variable air-gap attenuator was used to insert attenuation in the fiber link connecting the ODLs, and the bit-error rate was experimentally measured for different attenuator settings. Initial testing involved transmission of 2 k-byte packets of random data. For each attenuator setting, approximately 100,000,000,000 bits of data were transmitted through the link and verified using the TAXI digital link test system. Because of the amount of data involved, each test took approximately one week to complete.

Tables 1 and 2 give the results of the random data testing performed using the NEC and AT&T ODLs and the TAXI digital link test system. For the AT&T ODL devices, the attenuation shown is that inserted between the transmitter and the receiver, and it includes any losses in coupling cables. For the NEC device, the attenuation indicated is in addition to any loss associated with the 5 meter Fixed-Shroud Duplex (FSD) cable assembly used to connect the NEC ODL to the variable attenuator. It was impossible to measure the loss in this cable using the ST-type connector configuration on the available source and power meter. Fig. 8 and Fig. 9 are plots of the measured bit-error rate vs. attenuation for the NEC and AT&T ODLs, respectively.

The NEC NEOLINK-1312, according to the data sheet, has a typical power budget of 18.5 dB, and the AT&T ODL 200 pair has a typical power budget of 18.0 dB. The experimental results obtained here agree well with these stated typical values. The NEC NEOLINK-1312 has a worst-case power budget of 11.0 dB, according to the data sheet, while the AT&T ODL 200 pair has a worst-case power budget of 11.5 dB.

The codes used by the TAXI chip set do not have equal high and low times. This imbalance, commonly called a 'dc offset', can impact serial communication in several ways. Offset can affect ODL receivers by either causing a shift in the receive amplifier automatic gain control circuit or by limiting the

Attenuation (dB)	Bit-Error Rate
0.00	< 1 x 10**-11
0.70	< 1 x 10**-11
16.50	< 1 x 10**-11
16.70	1.8 x 10**-7
17.49	4.2 x 10**-6
18.02	6.9 x 10**-5

Table 1. NEC NEOLINK-1312 Bit-Error Rate Data

Attenuation (dB)	Bit-Error Rate
0.00	< 1 x 10**-11
0.56	< 1 x 10**-11
16.66	< 1 x 10**-11
16.92	1.9 x 10**-8
17.62	1.4 x 10**-6
18.02	6.9 x 10**-5

Table 2. AT&T ODL 200 Transmitter/Receiver Bit-Error Rate Data

Attenuation (dB)	Bit-Error Rate
0.00	< 1 x 10**-11
15.87	< 1 x 10**-11
16.87	1.7 x 10**-6

Table 3. AT&T ODL 200 Transmitter/Receiver Bit-Error Rate Data: Pathological Data (Long 0's - Long 1's Test)

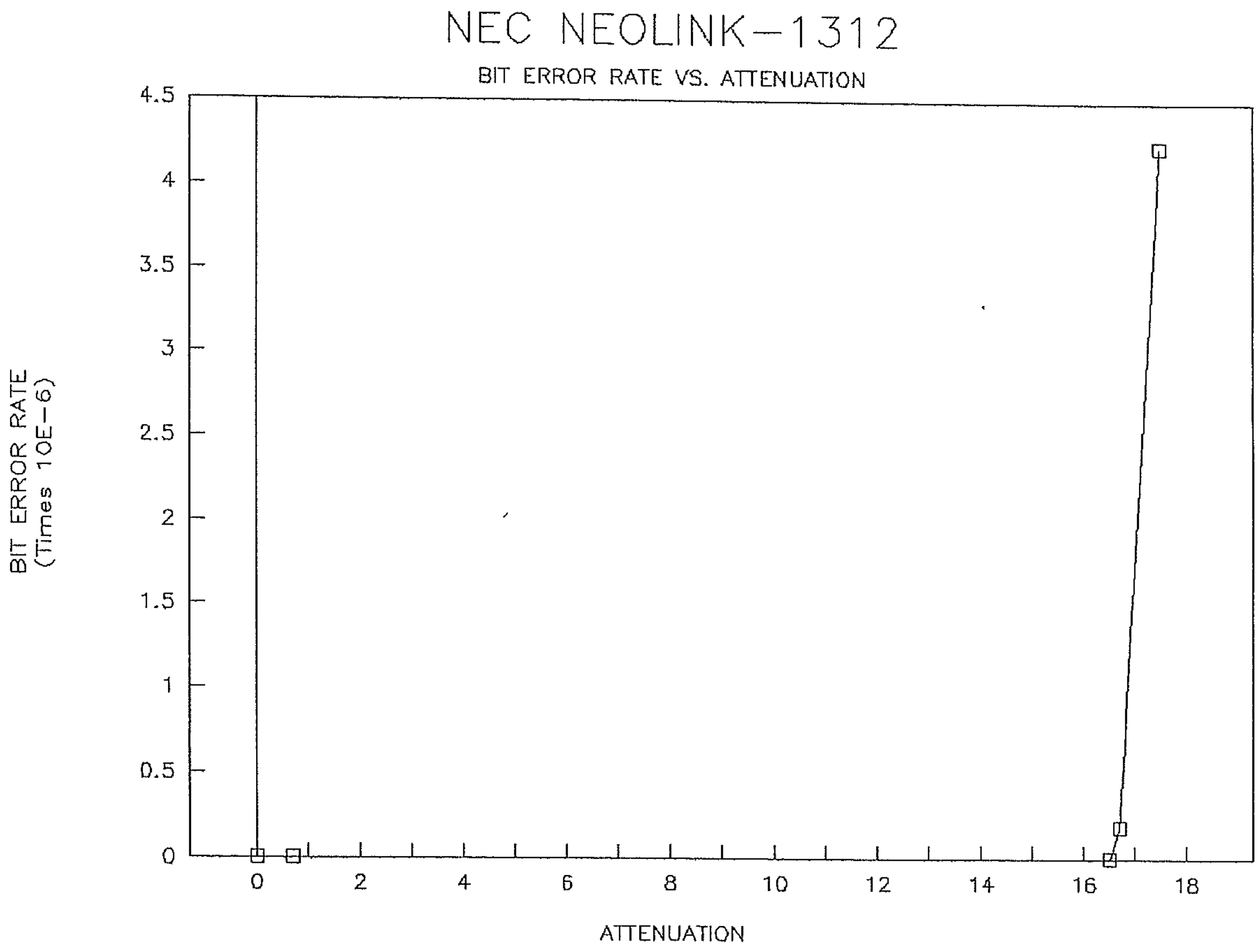

Fig. 8. Bit-error Rate vs. Attenuation for the NEC NEOLINK-1312 ODL

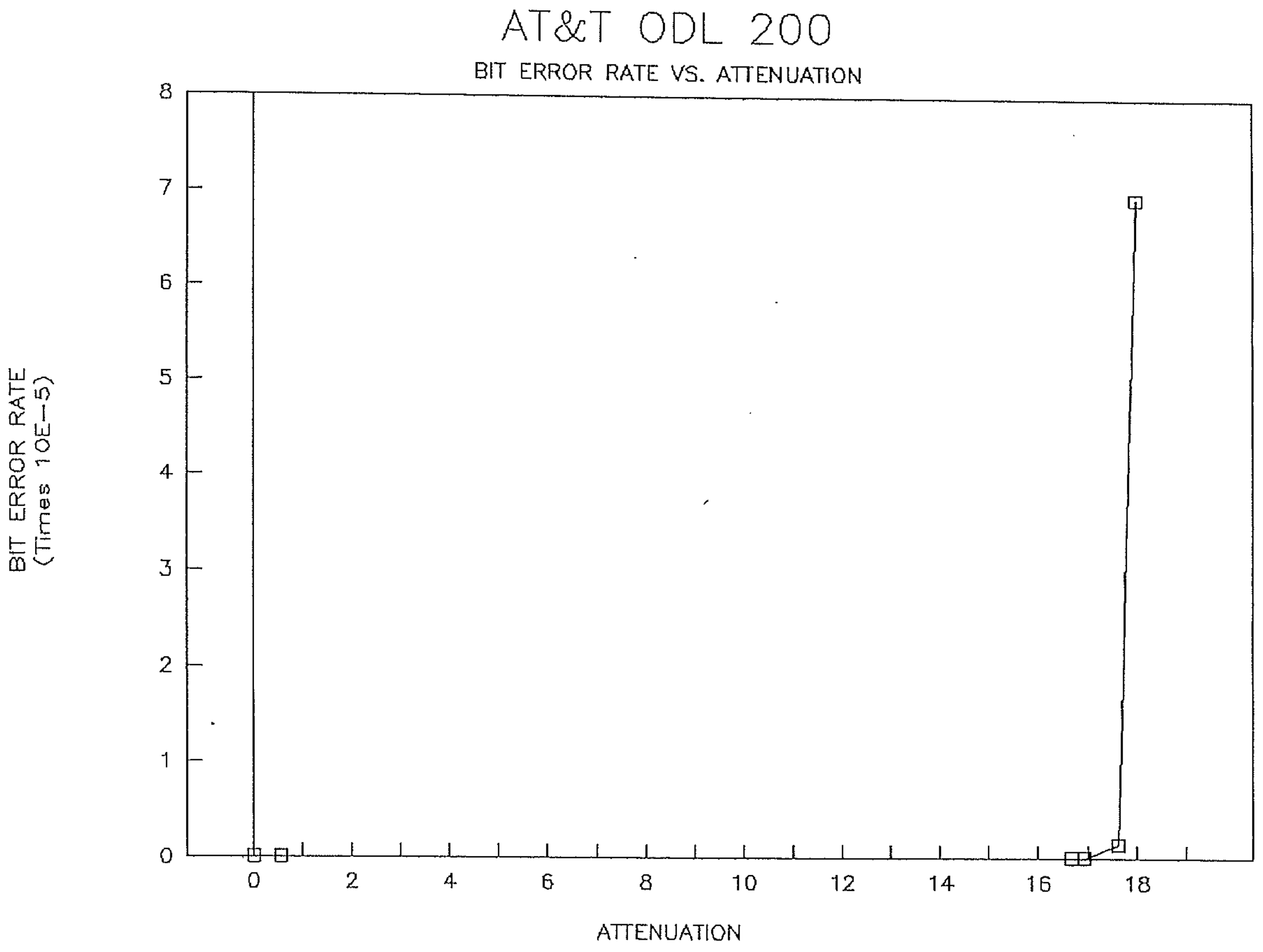

Fig. 8. Bit-error Rate vs. Attenuation for the AT&T ODL 200

minimum usable signal level. The dc offset associated with the codes used by the TAXI chip set can introduce jitter into the received data stream which must be removed by the receiving station. With this in mind, additional testing was performed on the AT&T ODL fiber link in an attempt to identify pathological data packets that would generate an increased bit-error rate due to the TAXI code imbalance discussed above. One such code, a long string of 0H nibbles followed by a long string of 1H nibbles, was identified. When encoded by the TAXI transmitter using the 4B/5B and NRZI encoding steps, each of these codes has a 20% dc offset, and the offsets are in opposite directions.

Table 3 gives the results of the pathological data test described above using the AT&T ODLs and the TAXI digital link test system. The actual test consisted of sending 1023 00H data bytes followed by 1023 11H data bytes. For this pathological case, the loss budget of the link was decreased by approximately 1 dB.

6 Conclusions

The digital link test system was used successfully to test the operation and performance of the Advanced Micro Devices TAXI chip set. The chip set was found to operate according to specifications, and metastability problems associated with the asynchronous transmitter handshake were not found. The TAXI chip set was found to interface very easily with commercial multi-mode ODLs from NEC and AT&T, and these devices, when used with the TAXI chip set, performed very well. The results of the testing described here indicate that a TAXI digital link will be very acceptable for use in the Fast Packet Switch.

References

[1] TAXIchip Integrated Circuits Technical Manual, Advanced Micro Devices, Sunnyvale, California, 1989.

[2] NEC NEOLINK-1312 DATA SHEET, NEC Corporation, Tokyo, Japan, 1989.

[3] AT&T ODL 200 Lightwave Data Link With Flag Data Sheet, AT&T Microelectronics, Allentown, Pennsylvania, 1988.

[4] Turbo Pascal V3.0 User's Guide, Borland International, Scotts Valley, California, 1986.

[5] Turbo C V2.0 User's Guide, Borland International, Scotts Valley, California, 1988.