Proceedings of the 1983
Winter Simulation Conference
S. Roberts, J. Banks, B. Schmeiser (eds.)

A NETWORK INTERFACE UNIT SIMULATION USING MICRO PASSIM

Tom P. Vayda
Larry L. Wear, Ph.D.
Department of Computer Science
California State University, Chico
Chico, CA 95929

This paper describes how micro PASSIM, a GPSS based simulation system, was transported from the Apple II to the HP 9836. The problems associated with moving a large program form one UCSD Pascal system to another are discussed. Micro PASSIM was transported to the HP system so that an Ethernet to HPIB interface board could be modeled. The model is described and the results obtained from the simulation are discussed. A discussion of the advantages and disadvantages of using micro PASSIM rather than a standard language, such as GPSS, is also included.

1. INTRODUCTION AND OBJECTIVES

This paper describes how the micro PASSIM a GPSS based simulation system, developed by Barnett, (Barnett 1981) was transported to the Hewlett-Packard 9836 desktop computer and used to simulate an Ethernet Network Interface Unit (NIU). PASSIM, which is based on GPSS, was developed at the University of British Columbia for use on a minicomputer (Uyeno 1980). PASSIM was modified by Claude C. Barnett, Walla Walla College, to run on the Apple II microcomputer. The system he developed, micro PASSIM, was written in the Apple version of UCSD Pascal.

The NIU described in this paper is designed to link the Hewlett-Packard Instrumentation Bus (HPIB) (Hewlett-Packard 1979), with Ethernet (Xerox 1980). The NIU has its own micro processor and buffer memory. It also has dedicated processors that interface with the HPIB and the Ethernet.

2. TRANSPORTING MICRO PASSIM TO THE HP 9836

As stated above, one of the reasons, we chose to use micro PASSIM was that it was written in UCSD Pascal and the HP 9836 supports UCSD Pascal. Unfortunately, this did not mean that programs could be transported directly from the Apple to the HP. The following describes the major differences between the two implementations.

2.1 Differences between Apple and HP Pascal

Some of the differences described in this section were eliminated by simply using the editor to replace one word with another. Others required rewriting some blocks of code or moving code from one module to another.

Both versions support string type variables; however, the HP compiler (Hewlett-Packard 1982) performs more checking on the length of the string when a constant or string variable is assigned to another string variable. This made it necessary to change the declarations on a number of variables and constants to satisfy the HP compiler.

For some reason, unknown to the authors, several Keywords differ between the two implementations. In the Apple version 'USES' (Apple 1980) indicates that a procedure will reference variables or procedures from a separately compiled module. Also, when variables or procedures within a module are to be made available to external procedures, they must be declared following an 'INTERFACE' statement. For the HP implementation 'USES' is replaced by 'IMPORT' (Hewlett-Packard 1982) and 'INTERFACE' is replaced by 'EXPORT'. In addition, HP uses 'IMPLEMENT' rather than 'IMPLEMENTATION' to specify the code section that has previously been declared in an 'EXPORT' block.

Apple Pascal has a feature which is not available on HP Pascal. In Apple Pascal, code can be written in a compound statement following the 'INTERFACE' block. This code is executed prior to the time the procedure is accessed by external routines. This makes is possible to include initialization code within the 'INTERFACE' block. Since this feature was not available in HP Pascal, some initialization code had to be moved to the main line procedure.

The text, The Pascal Handbook, (Tiberghien 1981) is a good reference to the differences between a number of implementations of Pascal.

2.2 Non-standard Pascal Features in Micro PASSIM

Some non-standard features of Apple Pascal had to be modified to run on the HP version. The routines to read the system clock, to access the random number generator, and to detect a key press fell into this category. All of these routines were rewritten for the HP implementation.

2.3 Extensions Available on the HP System

The HP implementation allows a larger range of values for integer type variables and has a much larger address space. This makes it possible to develop much more complex models on the HP system without encountering range problems with variables. The large address space eliminates the need to break procedures into small segments in order to get them to compile. It also allows more active transactions, more model segments, and more variables since these are all memory size dependent.

The problems encountered transporting programs from Apple UCSD Pascal to HP UCSD Pascal demonstrate the need for standardization in high level languages. Even two systems that are advertised as 'UCSD Pascal' have a significant number of differences that make the transporting of programs much more complicated than need be.

3. ADVANTAGES AND DISADVANTAGES OF MICRO PASSIM

Micro PASSIM has numerous features that make it a very useful simulation tool. Because it is interactive and easily extended to fit the application, it is very versatile. As we know, however, there is no such thing as a 'free lunch'. Both of these features come at the expense of increased program length and model development time.

3.1 Interactive Debugging with Micro PASSIM

The interactive nature of micro PASSIM makes debugging programs much easier than it is on batch systems using GPSS and most other common simulation languages. The ability to halt the simulation run at any point in time is very useful while debugging a model. Once the simulation has been suspended, information about the model and the accumulated statistics can be displayed. While the simulation is suspended, it is also possible to change the order in which transactions will be processed and the values of all model parameters. This can also be helpful in validating the performance of the model.

Debugging and validating the model is also simplified by the built-in 'Debug' facility. From the main menu the user can enter the Debug menu. This menu allows you to select the type of information you wish to have displayed while the simulation is running. At the '0' level no information is displayed. Levels 1 through 12 display information related to the execution of the simulation. Two features the authors found most useful were the single transaction trace, which displays relevant information about a transaction each time it changes state, and the display of each change of state by every transaction in the system.

3.2 Extending Micro PASSIM

Micro PASSIM can easily be extended by adding more Pascal procedures. We found this extremely valuable for two reasons; first, it was possible to include rather complicated logic within the model to alter the flow of transactions through the model (see Figure 4) and second, collecting statistics other than those predefined in the standard system was relatively easy.

3.3 Disadvantages of Using Micro PASSIM

The major disadvantages to using micro PASSIM rather than GPSS or some other simulation language are that it requires a great deal more code to develop a model and moderate to advanced programming ability in Pascal is required. For example, we developed a simple model that simulated the flow of jobs through a computer system which had multiple compilers, debuggers and run time cpu's. Implementation of the model required over 400 lines of Pascal code. The equivalent GPSS model (with fewer capabilities) required about 20 lines of code. The NIU described in this paper required over 1000 lines of code; a similar GPSS model, with fewer capabilities, required about 60 lines of GPSS code.

It must be pointed out, however, that the micro PASSIM simulation gave statistics that were not available in the GPSS model and part of the additional code was written to support the interactive features of micro PASSIM which are not available in GPSS. To help ease the pain of writing rather large Pascal programs when implementing a model in micro PASSIM, the author provided a module called a template. This module contains blocks of comments that show which procedures the users must write. It also includes the code necessary to initiate the simulation.

In summary, micro PASSIM models are longer and more difficult to write than those in GPSS but they can provide more information and are more convenient to modify and run.

4. DESCRIPTION OF THE NIU

In this section we describe the network interface unit (NIU) that is the subject of the simulation. The general function of the NIU is specified, followed by a description of its intended operating environment, an overview of its architecture, and then a description of how the NIU operates.

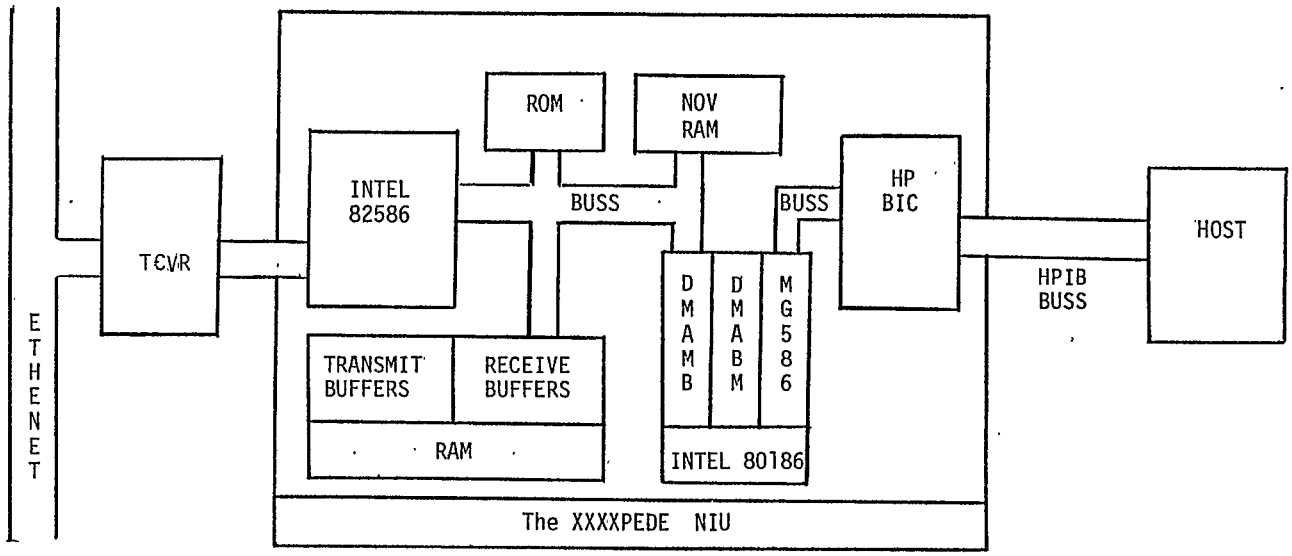

Figure 1: Network Interface Unit DIAGRAM

4.1 Function of the NIU

The main function of the NIU is to allow a host (DTE) to communicate with a local area network (LAN). The NIU provides layer 1 and 2 services of the ISO reference model for OSI (Tanenbaum 1981). Roughly speaking, the NIU has to pass packets (a stream of bytes delineated by a header and trailer) between the host and the LAN in both directions. To carry out this function the NIU has to provide buffer space for packets destined from the host to the LAN as well as packets that are addressed to the host coming from the LAN. This buffering must be transparent both to the host and the LAN, thus the NIU must have the intelligence to manage buffer memory. The NIU must provide error detection capability to ensure that faulty packets are not passed to the host. The NIU also has to recognize broadcast and multicast addresses, as well as to recognize and correctly respond to certain control packets.

4.2 Operating Environment

The host is assumed to be a Hewlett Packard (HP) minicomputer capable of fairly high speed data transfer (2M bits/sec) although any device capable of communicating via the Hewlett Packard Instrumentation Buss (HPIB) Protocol could serve as the host.

The LAN is assumed to be an Ethernet (Xerox 1980) capable of 10 M bits/sec data rate operating under the CSMA/CD control method (Xerox 1980). The NIU is not directly attached to the LAN. There is a standard Ethernet transceiver (TCVR) located between the Ethernet coaxial transmission cable and the NIU. The transceiver has the function of signal conversion as well as detecting the current state of the Ethernet and sensing other necessary electrical signals (Xerox 1980).

4.3 NIU Architecture

A block diagram of the major components of the NIU is given in Figure 1. The Intel 82586 is an intelligent local communication controller (LCC) (Intel 1983, Intel 1982a). It provides most of the functions of layers 1 and 2 of the OSI model including framing, error detection using the CCIT V.41 CRC polynomial, and single node, multicast, and broadcast addressing. The LCC uses a shared memory based architecture. The 82586 and the CPU (the 80186 in Figure 1) communicate through a shared memory (the RAM in Figure 1) using chained fixed sized buffers to store the inbound/outbound packets.

The Random Access Memory (RAM) is used as the buffer to store inbound/outbound packets until they are ready to be processed. It consists of a fixed size partition with one part storing packets bound from the host to the LAN (Transmit buffers) and the other part buffering packets being received from the LAN and bound for the host (Receive buffers).

The Backplane Interface Chip (BIC) is a standard HP communications product, capable of using the HPIB protocol. Its purpose is to receive/send packets from/to the host.

The read only memory (ROM) contains the firmware that controls the operation of the NIU. The non-volatile random access memory (NOVRAM) is used to store NIU configuration information. The ROM and NOVRAM are not part of the simulation.

The NIU contains two separate busses as shown in Figure 1: one is used for direct memory access (DMA) to/from the BIC, while the other is used to implement the shared memory architecture between the 82586 and the 80186 described above.

The Intel iAPX 186 (80186 in Figure 1) microprocessor (Intel 1982b) is used to control all other components of the NIU. It is a medium performance (8MHz) highly integrated 16 bit general purpose microprocessor. Its main functions include controlling the 82586 LCC and the shared memory space (RAM), (MG586), as well as carrying out direct memory access (DMA) transfers from the BIC to RAM (DMABM) and DMA from the RAM to the BIC (DMAMB). Its other functions are of no concern to the simulation.

The Ethernet transceiver and the Host are represented in the simulation simply as sources and sinks for the inbound and outbound packets to be processed by the NIU.

4.4 NIU Operation

In order to implement the functions described in Section 4.1, the NIU carries out the operations described below. We will consider only the two main functions of the NIU: transmit, transfer a packet from the host out over the LAN; and receive, transfer a correctly addressed, error-free packet received from the LAN to the host.

To transmit a packet, first the BIC sends an interrupt to the 80186, the interrupt service routine (ISR) in the 80186 initiates a DMA transfer to move the packet from the BIC to the transmit buffers (or tells the host to wait if no buffers are available), next the 80186 commands the 82586 to transmit this packet to the LAN (passing needed buffer address information), and finally the 82586 gets the packet from the transmit buffers and transmits it through the transceiver (TCVR) out over the LAN. See Figure 4 for details.

To receive a packet, first the 82586 interrupts the 80186 to notify the beginning of packet reception, the 80186 ISR responds by initiating a DMA transfer of the packet into the receive buffers. Next the 82586 logic checks the packet for errors and also to see if the whole packet was able to fit into the available receive buffers. If an error-free complete packet has been received, the 80186 is notified of this (via an interrupt) and the ISR initiates a DMA transfer from the receive buffers to the BIC, which in turn sends the packet to the host. If an incomplete or erroneous packet was received, the receive buffers are returned to the pool of free buffers and the packet is counted as a lost packet. This assumes that the layer 2 protocol ensures that the lost packet will be retransmitted by the sender after it times out for the lack of an acknowledgement.

5. DESCRIPTION OF THE NIU MODEL

In this section we describe the NIU simulation model. Note that the designers of the NIU have nicknamed it the XXXXPEDE (XXXX = centi or milli) thus explaining the model name on the output. We will discuss the purpose of the simulation, assumptions and limitations, an overview of the model, the transaction flow diagram, the model core, the model template, and finally describe the development effort required to implement the model.

5.1 Purpose

The decision to build a simulation model before the design phase of the XXXXPEDE had been completed was made for at least the following reasons:

1. to provide the designers with some design guidelines,
2. to provide an estimate of the performance to be expected from the XXXXPEDE (including lost packet statistics under various traffic loads),
3. to provide an estimate of component utilization (to see, for example, if the 80186 could be expected to perform additional functions), and
4. to provide specific information about the choice of memory (RAM) size and memory partitioning parameters.

5.2 Model Assumptions

Since many XXXXPEDE design choices were still undecided when the simulation model was designed, and to allow for a reasonable size first version of the model, the following assumptions were used during model construction:

1. The buffer memory (RAM) is to be subdivided into two dedicated parts: receive buffers and transmit buffers.
2. Each of the above mentioned memory partitions consists of a pool of fixed size buffers (not necessarily of the same size for the two portions).
3. In case of contention between receive and transmit packets at any of the XXXXPEDE resources, the receive packet is to receive higher priority (since the host is easier to 'choke off').
4. If a receive packet cannot be correctly processed because of insufficient buffer space, it is considered lost, and scheduled for retransmission.
5. If a transmit packet cannot be correctly processed because of insufficient buffer space, then the host will be 'choked off' (using HPIB flow control) and the packet will be queued in the host's buffers.
6. Various packet size and processing delay parameters were specified as constants.
7. The packet stream from both sides consists of a mixture of continuous (file transfer) transmissions or of 'random' (burst mode) transmissions consisting of a specified mixture of 'short' and 'long' packets. The packet stream from the LAN can have a different mixture, packet length, and distribution parameters than the packet stream coming from the host.
8. The internal busses of the XXXXPEDE are fast enough to handle all required DMA and control traffic, therefore bus contention does not have to be modeled.
9. The 80186 CPU is fast enough to handle its three main functions (manage the 82586, DMA from memory to BIC, DMA from BIC to memory) concurrently.

5.3 Limitations

The first version of the model (the subject of this paper) suffers from the following limitations:

1. There are no packets generated that contain errors.
2. The possible unavailability of Ethernet (channel contention) is not included.
3. The overhead for the recognition of multicast/broadcast addressed packets is not included in the model.
4. The possibility of packets that were transmitted from the Host to the LAN being lost is not included in the model.
5. The possibility of a collision and the subsequent use of the CSMA/CD exponential backoff algorithm is not included in the model.
6. There are other hardware/software details that are omitted from the model.

These limitations were acceptable because not enough hardware/firmware details were known during the model design phase, and also they allowed for an easier debugging/validation phase.

On the other hand, the model was designed so that future expansion to remove some of the above limitations could be easily incorporated into more refined versions of the model.

5.4 Model Overview

The model consists of two segments. One segment models the flow of transactions (packets) moving from the host to the LAN, while the other segment models the flow from the LAN to the host. Thus the model has two distinct transaction types which share some common storages.

The model time unit was decided to be a micro second.

The model uses the following storages (see Figure 1):

1. XC586 - the 82586 chip,
2. XMBUF - the transmit buffer space - capacity depends on memory size,
3. RCBUF - the receive buffer space - capacity depends on memory size,
4. BIC - the BIC chip,
5. DMAMB - the portion of the 80186 CPU controlling DMA from the RAM to the BIC,
6. DMABM - the portion of the 80186 CPU controlling DMA from the BIC to the RAM, and
7. MG586 - the portion of the 80186 CPU used to control the 82586 chip.

Note that our assumptions (Section 5.2) allow the 80186 to be modeled as three separate storages with dedicated functions. Also, our assumptions allow the omission of the XXXXPEDE internal busses from the storage list.

The major model parameters are memory size, fraction of memory for receive and transmit buffers, LAN and host data rates, retry delay for lost packets, and several variables controlling packet sizes and packet stream mix (different for the host and the LAN), and bandwidth utilization for both the host and LAN.

This list is not exhaustive. For a complete list of all model parameters, see Figure 3. During an interactive simulation run, the user is allowed to change the values of any or all of these parameters.

The packet stream is generated by a user defined function, according to the specifications described in Section 5.2. A nice feature of micro PASSIM is that it allows easy incorporation of user defined functions into the model.

The operation of the NIU was described in Section 4.4 in a cursory fashion. For a detailed description of how transactions move through the model, see the transaction flow diagrams in Figure 3.

The model core implements the transaction flow diagrams (Figure 3) in micro PASSIM. Figure 4 shows the model core. Note how Pascal statements are mixed with GPSS type blocks. It is this flexibility that makes micro PASSIM a powerful simulation tool.

In order to get a complete executable model, every section of the 'template' model has to be completed. The template contains the following sections:

User Globals - model parameters, storages, and auxiliary variables,

User defined functions and procedures,

Model Defaults - assigns reasonable values to all model parameters,

Model Menu - displays values of all model parameters and allows user to interactively change the value of any parameter,

Model Description - a help facility for the user's benefit,

Model Initialization - clears the model and initializes various model data structures,

Model Reset - allows user to reset simulation statistics without exiting the program,

Statistics - prints all model statistics including 'automatic' statistics (queue and storage statistics) as well as user defined statistics (Figure 5), and the

Model Core - described above (Figure 4).

```
.....MODEL PARAMETERS.....

                MODEL HARDWARE/SOFTWARE PARAMETERS
                ----- ---------- ---------

MEMORY SIZE                                 = 16384 BYTES
FRACTION OF MEMORY FOR  RECEIVE BUFFERS = 0.875
FRACTION OF MEMORY FOR TRANSMIT BUFFERS = 0.125
THERE ARE 56  RECEIVE BUFFERS OF 256 BYTES EACH
THERE ARE  8 TRANSMIT BUFFERS OF 256 BYTES EACH

ETHERNET DATA RATE = 1.25 BYTES/MICSEC
BIC(HOST) DATA RATE= 0.25 BYTES/MICSEC

RETRY DELAY FOR LOST PACKETS (ENET) 20000

TIME TO SET UP BIC TRANSMISSION     =  100
TIME TO SET UP ENET TRANSMISSION    =  100
TIME TO ACQUIRE BIC                 =  200
TIME TO PROCESS HOST INTERRUPT      =   10
TIME TO PROCESS ETHERNET INTERRUPT  =   10
TIME TO SET UP BIC DMA              =   50
TIME TO FINISH BIC DMA              =   50
TIME TO SET UP 586 DMA              =    5
TIME TO FINISH 586 DMA              =   50
TIME TO ACQUIRE TRANSMIT BUFFER     =  150
TIME TO RELEASE TRANSMIT BUFFER     =  150
TIME TO RELEASE RECEIVE  BUFFER     =  150
ETHERNET RECEIVE PRIORITY           =    2
BIC(HOST) TRANSMIT PRIORITY         =    1

                MODEL TRAFFIC PARAMETERS
                ----- ------- ----------

ETHERNET  PACKET SIZE(BYTES)   106 MINIMUM  1500 MAXIMUM
BIC(HOST) PACKET SIZE(BYTES)   106 MINIMUM   256 MAXIMUM

RATIO OF CONTINUOUS MODE/BURST MODE PACKETS     =  0.500
RATIO OF LONG/SHORT PACKETS IN BURST MODE       =  0.250
AVERAGE LENGTH OF SHORT PACKET FROM ETHERNET    =  106.0
AVERAGE LENGTH OF SHORT PACKET FROM BIC(HOST)   =  106.0
AVERAGE # OF PACKETS IN CONTINUOUS MODE ON ETHERNET =  6.0
AVERAGE # OF PACKETS IN CONTINUOUS MODE ON BIC(HOST)=  4.0

AVERAGE NUMBER OF BYTES PER PACKET FROM ETHERNET.  =  867.4
AVERAGE NUMBER OF BYTES PER PACKET FROM BIC(HOST)  =  165.0
AVERAGE PACKET INTERARRIVAL TIME FROM ETHERNET     = 2313.1
AVERAGE PACKET INTERARRIVAL TIME FROM BIC(HOST)    = 2200.0

NUMBER OF ACTIVE NODES ON ETHERNET              =    1
ETHERNET TOTAL UTILIZATION                      =  0.300
XXXXXPEDE'S UTILIZATION OF ETHERNET XFER CAPACITY =  0.300
XXXXXPEDE'S UTILIZATION OF BIC(HOST) XFER CAPACITY =  0.300

                MODEL DEBUG PARAMETERS
                ----- ----- ----------

          PRINT PACKET INFORMATION  : FALSE
          PRINT BUFFER INFORMATION  : TRUE
          PRINT ALL INFO (IN STATS) : TRUE
```

FIGURE 2: MODEL PARAMETERS

The complete implementation of the model required about 1000 lines of Pascal code, including comments. This does not include the library modules of the micro PASSIM system that have to be linked to the executable (compiled) form of the model. The complete design and implementation and validation of the model required about 80-100 hours.

6. RESULTS OF NETWORK INTERFACE UNIT SIMULATION

This section presents a description of the statistics that were gathered when the model was run and a display of some of the data that was gathered from numerous runs of the simulation.

6.1 Micro PASSIM Result Printout

The accumulated statistics for the current run of the simulation can be displayed or printed whenever the micro PASSIM main menu is displayed on the screen. The printout for a typical run is shown in Figure 5. Some of this information is similar to what you would see from a GPSS simulation. The QUEUE statistics and STORAGE utilization fall into this category.

The portion of the output labeled 'PACKET STATISTICS', however is unique to the model we designed. The statistics displayed here are ones that we felt would be useful for our purposes. We were responsible for generating the statistics and for displaying them in the format shown.

6.2 Model Performance and Bus Utilization

Figure 6 shows how the model responded to various levels of Ethernet utilization. In this figure, we chose to plot number of packets that had to be retransmitted, (Lost Packets), total data throughput, (Data Rate), and utilization of transmit and receive buffers, XMBUF, and RCBUF. For this figure, the buffer memory size was 8192 bytes, the BIC utilization was 0.40 and the ratio of receive to transmit buffers was 7:1. The maximum data rate for the BIC was 2 M bits/sec and for the Ethernet was 10 M bits/sec for all the cases described in this section.

The figure shows one result which was expected; as the data rate of incoming packets from Ethernet approaches the maximum rate at which data can be transmitted to the BIC, the number of lost packets increases drastically. It also shows that the transmit buffers are utilized much more heavily than the receive buffers. This is due in part, to the fact that emptying receive buffers was given higher priority than emptying transmit buffers and therefore data tended to remain in transmit buffers for a much longer period of time.

The next graph, Figure 7, shows how the model responded to changes in incoming packet rates from the host, BIC utilization. This term is somewhat misleading because the NIU can halt the actual transmission of data from the host computer through the BIC by sending a 'not available' back to the host. Since this transmission from the host to the LAN can be choked off by the NIU, there is no significant increase in lost packed count and only a small increase in receive buffer utilization while transmit buffers utilization increases to nearly 100 percent. Total data rate also increases and approaches the maximum of the BIC.

6.3 Model Performance and Buffer Memory Size

Figure 8 shows how the NIU performs with various amounts of buffer memory from a minimum of 4096 bytes to a maximum of 32,768 bytes. For this figure the Ethernet utilization was set to 0.18 and the BIC utilization to 0.40. Two of the more interesting results that can be seen are that the data transfer rate remains constant and there is a linear decrease in receive buffer utilization as memory size increases.

6.4 Execution Time of Model

The execution speed of the simulation language is often quite important if a model is going to be run many times. On the HP 9836, each run of 3 seconds of simulated time required about two minutes of real time. In a typical run about 650 packets were transmitted from Ethernet and 1750 requests were generated from the Host. The resolution of the simulation clock was 1 micro second and the length of most of the ADVANCE's

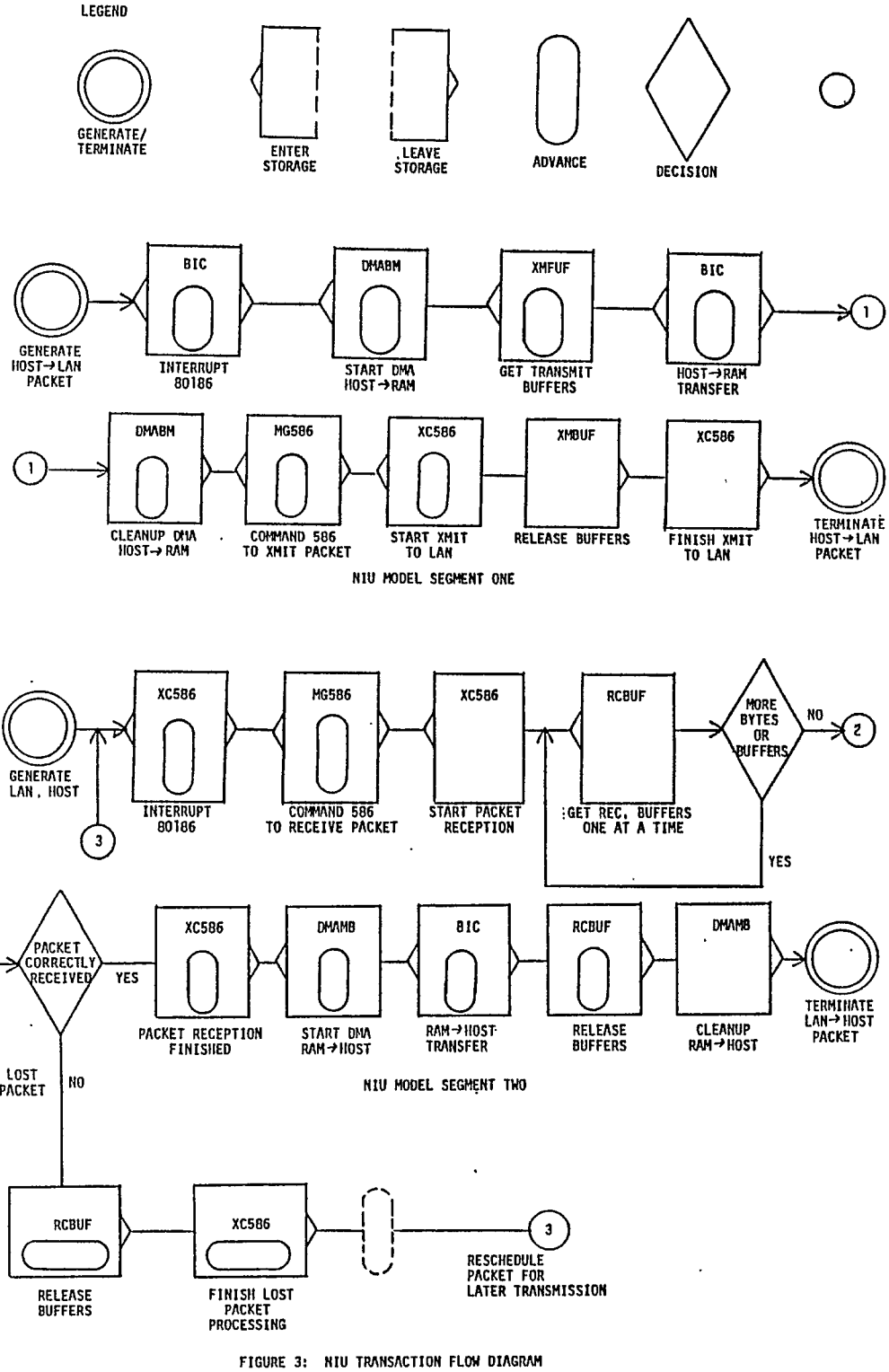

FIGURE 3: NIU TRANSACTION FLOW DIAGRAM

```
(*                MODEL SEGMENT ONE  (HOST/BIC)                    *)

START: BEGIN
         GENERATE(02,PACKGEN(HP),XMPRIORITY,SIMCLOCK<CLOSE_TIME);
         PAR[3] := PACKETSIZE;   (*  PACKET SIZE ASSIGNED BY PACKGEN *)
         PAR[4] := PAR[3] DIV XMBSIZE;  (* NUMBER OF BUFFERS REQUIRED *)
         REM := PAR[3] MOD XMBSIZE;
         IF REM>0 THEN PAR[4] := PAR[4] + 1;
       END;
   02: ENTER(03,1,BIC);
   03: ADVANCE(04,HOST_INTRUPT);
   04: LEAVE(05,1,BIC);
   05: ENTER(06,PAR[4],XMBUF);
   06: ADVANCE(07,ACQXMBUF );
   07: ENTER(08,1,DMABM);
   08: ADVANCE(09,SETUPBICDMA);
   09: ENTER(10,1,BIC);
   10: ADVANCE(11, PAR[3] / DATA_RATE[HP] );
   11: LEAVE(12,1,BIC);
   12: ADVANCE(13,FINISHBICDMA);
   13: LEAVE(14,1,DMABM);
   14: ENTER(15,1,MG586);
   15: ADVANCE(16,SETUPENXM);
   16: LEAVE(17,1,MG586);
   17: ENTER(18,1,XC586);
   18: ADVANCE(19,PAR[3]/DATA_RATE[EN]);
   19: LEAVE(20,1,XC586);
   20: ADVANCE(21,RELXMBUF);
   21: LEAVE(22,PAR[4],XMBUF);
   22: BEGIN
         TOTBYTES_HP := TOTBYTES_HP + PAR[3];
         TOTPACS_HP  := TOTPACS_HP + 1;
         TERMINATE(1);
       END;


(*          MODEL SEGMENT TWO (ETHERNET)                    *)

START2: BEGIN
          GENERATE(52,PACKGEN(EN),RCPRIORITY,SIMCLOCK<CLOSE_TIME);
          PAR[3] := PACKETSIZE;    (*  PACKET SIZE ASSIGNED BY PACKGEN *
          PAR[4] := PAR[3] DIV RCBSIZE;  (* NUMBER OF BUFFERS REQUIRED *
          PAR[5] := 0;    (* COUNTS NUMBER OF RETRIES TO RECEIVE PACKET *
          PAR[6] := 0;  (* COUNTS NUMBER OF BUFFERS ACQUIRED BY PACKET *
          REM := PAR[3] MOD RCBSIZE;
          IF REM>0 THEN PAR[4] := PAR[4] + 1;
          NX_BLOCK := 52
        END;
    52: ENTER(53,1,XC586);
    53: ADVANCE(54,ENET_INTRUPT);
    54: LEAVE(55,1,XC586);
    55: ENTER(56,1,MG586);
    56: ADVANCE(57,SETUP586DMA);
    57: LEAVE(58,1,MG586);
    58: ENTER(59,1,XC586);
    59: BEGIN
          MOREBYTES := TRUE;
          WHILE ( RCBUFCNT > 0 )  AND  MOREBYTES DO
            BEGIN
              ENTER(60,1,RCBUF);
              RCBUFCNT := RCBUFCNT - 1;
              PAR[6] := PAR[6] + 1; (* COUNT CAPTURED BUFFERS *)
              MOREBYTES := ( PAR[4] > PAR[6] );
            END;
          ADVANCE(60,PAR[3]/DATA_RATE[EN]);
        END;
    60: BEGIN
          ADVANCE(66,FINISH586DMA);
          IF PAR[4] > PAR[6]  (* BUFFERS REQUIRED > BUFFERS ACQUIRED? *)
            THEN  NX_BLOCK := 61;  (* THE PACKET WAS LOST *)
        END;

(* HANDLE A LOST PACKET *)
61: ADVANCE(62,RELRCBUF);
62: BEGIN
      RCBUFCNT := RCBUFCNT + PAR[6];
      LEAVE(64,PAR[6],RCBUF);
      PAR[5]  := PAR[5] + 1;      (* COUNT TIMES PACKET SENT ON ENET *
      PAR[6]  := 0;               (* RESET BUFFERS ACQUIRED COUNTER  *
      LOSTBYTES := LOSTBYTES + PAR[3]; (* COUNT NUMBER OF LOST BYTES*
      LOSTPAC   := LOSTPAC + 1;     (* COUNT NUMBER OF LOST PACKETS  *
    END;
64: LEAVE(65,1,XC586);
(* RESCHEDULE IT FOR LATER ETHERNET TRANSMISSION *)
65: SCHEDULE(52,RCPRIORITY,SIMCLOCK+RETRYDELAY,CUR);

(* HANDLE A CORRECTLY RECEIVED PACKET *)
66: BEGIN
      IF PAR[5] = 0

        THEN  (* PACKET WAS RECEIVED ON FIRST TRY *)
          GOT_THROUGH_FIRSTTRY := GOT_THROUGH_FIRSTTRY + 1

        ELSE  (* PACKET WAS SENT MORE THAN ONCE BEFORE RECEPTION *)
          BEGIN
            RETRIEDPACKS := RETRIEDPACKS + 1;
            RETRYTOT     := RETRYTOT + PAR[5];
          END;

      LEAVE(67,1,XC586);

    END;
67: ENTER(68,1,DMAMB);
68: ADVANCE(69,SETUPBICDMA);
69: ENTER(70,1,BIC);
70: BEGIN
      HPXM_DELAY := SETUPHPXM  + (* TIME TO 7 *)
                    HPBUSDELAY + (* TIME TO 7 *)
                    PAR[3] / DATA_RATE[HP]; (* TIME TO XFER DATA *)
      ADVANCE(71,HPXM_DELAY);
    END;
71: LEAVE(72,1,BIC);
72: ADVANCE(73,FINISHBICDMA);
73: LEAVE(74,1,DMAMB);
74: ADVANCE(75,RELRCBUF);
75: LEAVE(76,PAR[4],RCBUF);
76: BEGIN
      RCBUFCNT    := RCBUFCNT + PAR[4];
      TOTBYTES_EN := TOTBYTES_EN + PAR[3];
      TOTPACS_EN  := TOTPACS_EN + 1;
      TERMINATE(1);
    END;
```

Figure 4: Micro PASSIM Model Core

was between 5 and 150 microseconds. Because of the relatively short run times, it was feasible to make numerous changes to the model to examine the response under many different conditions.

7. SUMMARY

This paper described the HP 9836 implementation of micro PASSIM and its application to modeling an NIU. To implement the model, first the authors had to transport the micro PASSIM system from Apple II to HP9836 in order to be able to use the 1 megabyte memory capacity of the HP 9836.

While the current version of the model has certain limitations, even in this form it has proven to be a useful and effective model of the XXXXPEDE NIU. The authors are currently expanding the model to remove some of the limitations described in Section 5.3.

The micro PASSIM system has already proven itself to be a powerful simulation tool, but it does have room for improvement. Some of the desirable expansions include graphics output, file input of various model configurations, the ability to run the system using command files, and the implementation of more GPSS blocks. The author of the micro PASSIM system is currently working on several of these enhancements.

In conclusion, the authors have found micro PASSIM to be a powerful, interactive, efficient, easy to customize, and 'infinitely' expandable simulation tool.

```
----------S I M U L A T I O N  S T A T I S T I C S----------

                HP 98xx uPASSIM P1.0
          XXXXXPEDE SIMULATION MODEL V2.0
                  September 1983

              SIMULATION  HALTED
                AT STOP TIME

TERMS REL:    2074                        SIMCLOCK ABS:3000000.00
TERMS MAX:   50000                        SIMCLOCK REL:3000000.00
XACTS NOW:     386                           STOP  REL:3000000.00
XACTS MAX:     888                          CLOSE REL:3000000.00
SEED  RND:      17                           SEED  SET:    0.00
                        REAL TIME:12169.00

---------- QUEUE STATISTICS ----------

   NAME      CURRENT   AVERAGE   AVERAGE   NUM OF    ZERO     MAXIMUM    MAXIMUM
             CONTENTS  CONTENTS    WAIT     EXITS    WAITS    CONTENTS     WAIT

BICCHIP Q        1       3.94   3089.57     3829      723        16     13552.14
586CHIP Q        0       0.02     22.36     2779     2591         3      1141.10
586MNGR Q        0       0.00      0.62     2075     2050         1        97.10
DMA_M_B Q        0       3.30  14063.18      704      153        15     55227.79
DMA_B_M Q        7       6.67  14578.82     1372        3         7     50088.00
RBUFMGR Q        0       0.00      0.00     2113     2113         0         0.00
XBUFMGR Q      374     181.38 394586.91     1379       18       389    642974.08

---------- STORAGE STATISTICS ----------

   NAME      CURRENT   AVERAGE   AVERAGE   NUM OF  UTILIZATION MAXIMUM   CAPACITY
             CONTENTS  CONTENTS TIME/XACT  EXITS               CONTENTS

BICCHIP          1       0.98    766.88     3828      0.98        1          1
536CHIP          0       0.19    210.01     2779      0.19        1          1
586MNGR          0       0.05     67.77     2075      0.05        1          1
DMA_M_B          0       0.82   3515.46      704      0.82        1          1
DMA_B_M          1       1.00   2187.19     1371      1.00        1          1
RBUFMGR          1      15.41   4267.43      703      0.28       56         56
XBUFMGR          8       7.93   2188.18     1371      0.99        8          8

                      PACKET STATISTICS
                      ------ ----------

                          PACKETS GENERATED
                  FROM ETHERNET          FROM HOST
PACKET TYPE
CONTINUOUS        236  33.5%              485  27.7%
LONG BURST        132  18.8%              355  20.2%
SHORT BURST       336  47.7%              914  52.1%
                  ----                    ----
      TOTALS      704  28.6%             1754  71.4%

TOTAL NUMBER OF BYTES GENERATED     :  736137 = 2458 PACKETS - AVG SIZE = 299.5
TOTAL BYTES SENT BY THE HOST        :  213639 = 1371 PACKETS - AVG SIZE = 155.8
TOTAL BYTES RECEIVED FROM ETHERNET  :  462974 =  703 PACKETS - AVG SIZE = 658.6
NUM. OF PKTS. RECEIVED ON FIRST TRY :  704 100.0%  OF PACKETS SENT ON ETHERNET
EFFECTIVE DATA TRANSFER RATE        :  225538 BYTES/SEC


NUMBER OF RECEIVE  BUFFERS          :  56 OF 256 BYTES EACH = 14336 TOTAL BYTES
NUMBER OF TRANSMIT BUFFERS          :   8 OF 256 BYTES EACH =  2048 TOTAL BYTES
```

Figure 5: Sample Output from Micro PASSIM Simulation

REFERENCES

Barnett CC (1981), micro PASSIM: A discrete-Event Simulation Package For A Microprocessor Using UCSD PASCAL. IN: Modeling and Simulation On Microcomputers, LA Leventhal (ed), Society For Computer Simulation, San Diego, CA, pp 60-64.

Hewlett-Packard (1974), The Hewlett-Packard Interface Bus - HP-IB, Hewlett-Packard Company, Palo Alto, CA.

Hewlett-Packard (1981), Pascal Language System User's Manual for the HP9826 and HP 9836 Computers, Hewlett-Packard Company, Fort Collins, CO.

Intel (1982a), The Complete VLSI LAN Solution, Intel Corporation, Sunnyvale, CA.

Intel (1982b), iAPX186 High Integration 16-bit Microprocessor, Intel Corporation, Sunnyvale, CA.

Intel (1983), 82586 Reference Manual, Intel Corporation, Sunnyvale, CA.

Tanenbaum AS (1981), Computer Networks, Prentice-Hall, Englewood Cliffs, NJ.

Tiberghein J, (1981), The Pascal Handbook, Sybek Inc., Berkeley, CA.

Uyeno DH, Vaessen W (1980), PASSIM: A Discrete-Event Simulation Package for PASCAL. IN: Simulation, Vol 35, No 6, pp 183-190.

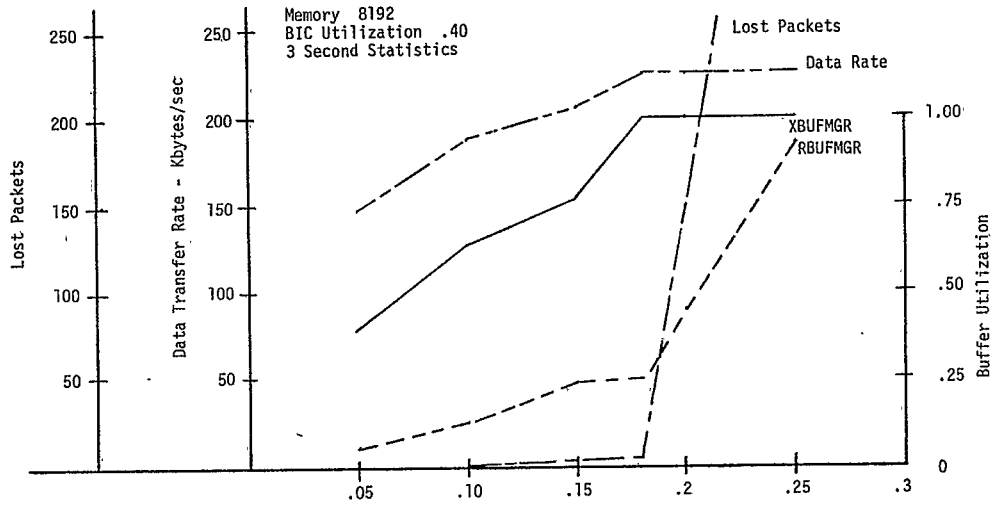

Figure 6: Performance as a Function of Ethernet Utilization

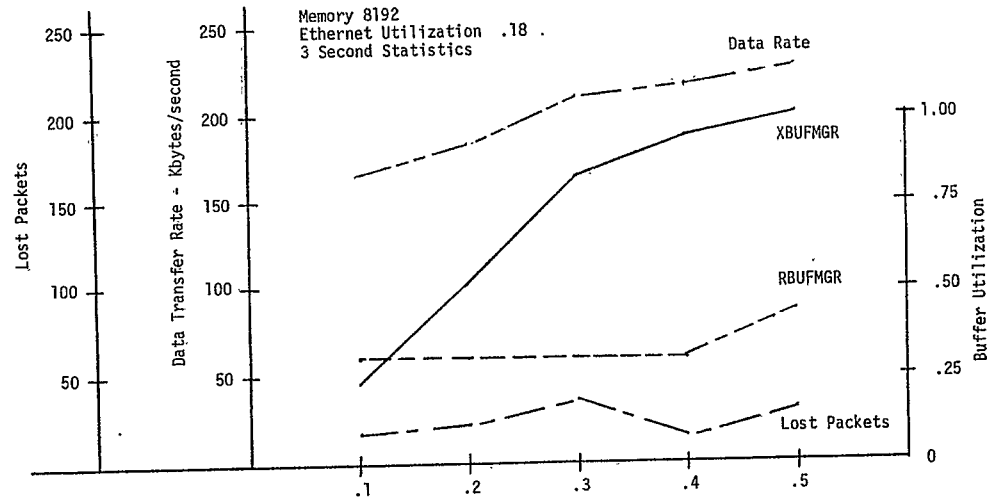

Figure 7: Performance as a Function of BIC Utilization

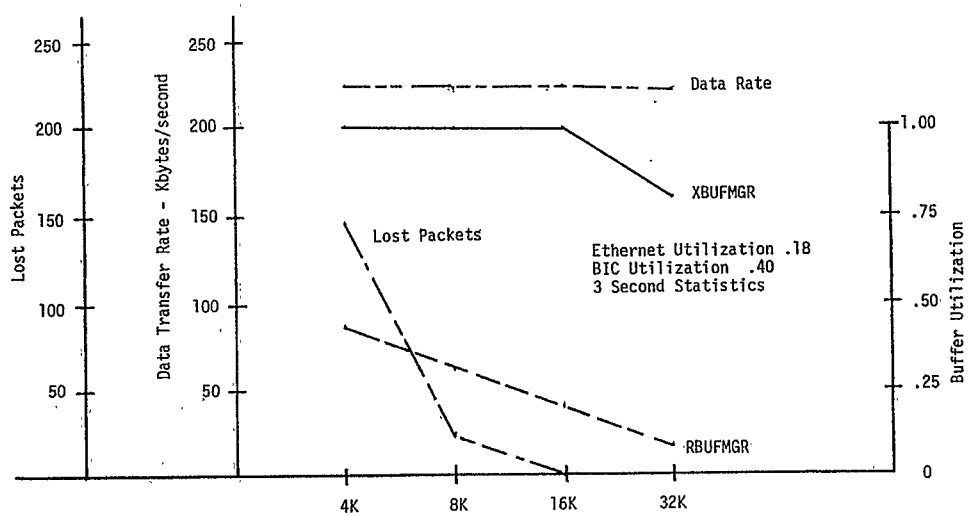

Figure 8: Performance as a Function of Memory Size