# THE GOULD NP1 SYSTEM INTERCONNECTION

**Duc J. Vianney, James H. Thomas, Vicki Rabaza**

Gould Inc., Computer Systems Division
6901 W. Sunrise Blvd., Ft. Lauderdale, FL. 33313-4499

## ABSTRACT

**The Gould NP1 is a multicomputer multiprocessing system designed for high performance and parallel processing required in diverse scientific and engineering applications. The NP1's basic building block is a dual-processor single bus system which can be expanded up to eight processors over four system buses.**

**This paper discusses the overall design and implementation of the NP1 system interconnection in particularly the inter-system bus link which interconnects four system buses to form a tighly coupled eight processor system. Such interconnectivity provides NP1 the flexibility in system expansion capable of addressing full four gigabytes of physical memory with the least communications delay.**

**Index Terms -** Gould NP1, dual-cpu, multiprocessor, processor farm, inter-system bus link.

## 1. INTRODUCTION

In a multicomputer organization, each computer is considered as a basic computing module. A module consists of a CPU, local memory, and I/O subsystems. All modules are interconnected through a low-speed communication schemes of the direct-switching or store-forward type. As a result, infrequent data exchanges among the various computers in the system are expected. This approach speeds up operations by matching the computer network topology to that of the complex application. Since each computer executes independent instruction streams with little interaction, this configuration implements mainly instruction parallelism but no data parallelism [1]. An immediate drawback of such an organization is that there is no direct computer intermodule communication between the processor and memory modules of different computers.

In a multiprocessor architecture, several processors operate in parallel in an asynchronous manner. This corresponds to parallelism at the functional module level where all modules are interconnected to exchange data directly with each other through a fast interconnection mechanism. The direct intermodule communication minimizes idle processor time by increasing program parallelism. There are two basic approaches used in direct interconnection: shared bus or shared memory. One approach, e.g., the partitioned multiprocessor, provides shared global bus and nonshared local memories [2]. The other alternative uses shared memory [3, 18], or a hybrid of both shared bus and shared memory [4].

While partitioned architectures offer a high degree of efficiency in bus utilization, the overhead of task assignment, load balancing and synchronization can be high [5, 6]. The advantage of shared memory is in the access of semaphores protecting shared data required to schedule and synchronize processes in contrast to such disadvantages as memory contention, lockouts for critical sections, scheduling complexity, and communication delays [7, 8].

The Gould NP1 system is organized in a rather unique fashion [9, 10, 11]. One of its design goals is configurability. The system has to be flexible in its expansion capability, from a single processor to eight processor system, without major interconnection delay amongst the processors while still has to maintain its high aggregate processing performance. In addition, each configuration should be capable to address large physical memory, a total of four gigabytes for a fully configured system. Since the physical memory in NP1 is seen contiguously by all processes, accesses to shared data and synchronizing semaphores have to be efficient. To resolve these issues, NP1 is designed around a system bus. Each system bus supports either a single or dual CPUs, a maximum of one gigabyte of memory, and various I/O subsystems. As the system expands, direct connection between system buses is provided through the Inter-System Bus Link (ISBL). There is an ISBL for each pair of system buses, thus coalescing them into a sort of global bus and providing two gigabytes of memory to processes that run on either system bus. Similar connection strategy is also used for four system bus expansion to achieve maximum system capacity.

In this paper, the design, implementation and some issues related to the system interconnection are discussed. An overview of the Gould NP1 system will be first given followed by description of the system bus, ISBL and its features.

## 2. SYSTEM OVERVIEW

### The Gould NP1 System

NP1 is the first member of the Gould NPL mini-supercomputer family. The hardware characteristics can be summarized as follows:

- A high speed system bus with 154 megabytes transfer rate.
- Two CPUs per system bus, 52nsec clock cycle.
- Four system buses and eight CPUs in full system configuration.
- 64 bits wide data transfer on each system bus in parallel with command and control transfers.
- Optional Arithmetic Accelerators to enhance the performance of scalar and vector floating-point arithmetic operations.
- One gigabyte of physical main memory per system bus for a maximum of four gigabytes per system.
- 32K-Byte cache per CPU partitioned into 16K-Byte for data and 16K-Byte for instruction.
- Independent I/O processors with high speed I/O bus.
- 154MB/sec data transfer across system buses.

### System Organization

Each NP1 system bus contains a number of units, nucleus and I/O. The nucleus units are connected directly to the system bus. They include the Central Processing Units (CPUs), memories, a Bus Controller (K-Board), a Bus SnapShot (S-Board), Inter-System Bus Links (ISBLs), and Universal I/O Microengines (UIOMs); together with the optional Arithmetic Accelerators (AAs).

**CPU.** The NP1 CPU is a five board set with an optional two board set arithmetic accelerator (AA). It is an ECL based machine, making extensive use of LSI, VLSI, and custom gate array technology. The CPU comprises three functional units: the Instruction Unit (I-Unit), the Execution Unit (E-Unit), and the Microstore/Instrumentation Unit (MX-Unit).

The MX-Unit provides both the microstore function and the instrumentation function. The microstore is 8K words deep by 100 bits wide, soft loadable from the Control Unit (an element of the Bus Controller) upon power up thus facilitating microcode update and debugging tasks. The instrumentation function includes clock control, dynamic snapshot, static register and bus display, software counters and interval timer, and operator control panel functions for the CPU. The dynamic snapshot captures the contents of major CPU buses during every clock cycle while the CPU running. The software controlled counters are used to measure three performance parameters: 1) the ratio between cache hits and memory accesses, 2) the translation buffer hit/miss ratio, and 3) the instruction and operand cache accesses. The software can clear and read these counters and all data are visible to the user.

The E-Unit consists of the hardware necessary to execute logical and arithmetic operations. It handles fixed-point and floating-point numbers, logical operands, and is especially equipped to process multiple element vector arrays. It provides static display visibility for itself and the physical address bus. It also maintains the Trap Register to report error conditions and the Base Registers for address calculations. There are other trap registers in the MX-Unit and I-Unit as well.

The I-Unit performs instruction prefetch, instruction pipeline, instruction decode, operand address calculation, and register file addressing. It also handles program counter update, branch logic, address translation mechanism, etc.

**I/O Structure.** The I/O modules are connected to NP1 via one of the Intelligent Peripheral Interface (IPI) buses off the UIOM. The IPI bus is an industry standard bus which allows the connection on that bus of different classes of devices with diverse speeds. Each IPI bus can support up to eight I/O modules, including the Disk Intelligent Modules (DIM), the Communication Processor Intelligent Module (CPIM), and the VME interface module. The DIM is capable of supporting up to four high speed disk devices. With eight communications ports that can be configured independently, the CPIM supports a host of communication protocols including bit synchronous, byte synchronous, IBM bisync, and asynchronous. The VME interface module supports a VMEbus based system which is itself a complete microprocessor system with tape, line printer, terminal cluster and communications subsystems. Figure 1 illustrates a typical organization of an NP1 system.

**Bus Controller.** The Bus Controller performs the traditional functions of a bus controller such as bus arbitration and system clock control. In addition, the K-Board includes an on-board microprocessor system called the system Control Unit. The Control Unit (CU) has its own winchester disk which contains all system microcodes, diagnostics, and various application software. The CU can be considered as the command and control center of the NP1 system. It monitors the status of the nucleus units via the Serial Link Gate Array Bus (SLGA). Through the SLGA, the CU can also download or upload the module microcode, turn a module on- or off-line, request the module to execute its onboard microdiagnostics, etc. The CU communicates all system data to the outside world via the system primary station. The primary station provides a menu driven user interface from which the operator can monitor the system status, maintain the CU software and NP1 microcode, perform system configuration, power up or down, and execute system diagnostics. Remote access to NP1 is also controlled by the primary station. The monitoring of environment conditions such as power supplies and temperature is performed by the Supervisory Board which periodically reports its findings to the user through the primary station.

## 3. NP1 - A PARALLEL MULTIPROCESSING SYSTEM

In general, the underlying architecture of a parallel processing system is identified by its granularity and the interconnection of its processing elements.

Granularity refers to the speed and capacity of an individual processing element [12]. Viewing the system as a whole, the speed is directly proportional to the number of processors available within the system. Thus, granularity also refers to the size and number of processors in a parallel machine environment. A small-grained or fine-grained system has a large population of slow processors (or possibly processor/memory pair) that yields a much higher aggregate performance. A typical small-grained system usually consists of 16 up to 64K processors (e.g., FPS T-Series), and a front-end processor. In contrast, a large-grained or coarse-grained system has a small number (usually 2 to 16) of large processors, each of which is relatively fast and may have its own I/O subsystem.

The other characteristic of parallel processing machine involves the linkage between processing elements, i.e., the connection and communication between processing elements. Depending upon the implementation strategy, they can be simply classified as **processor**

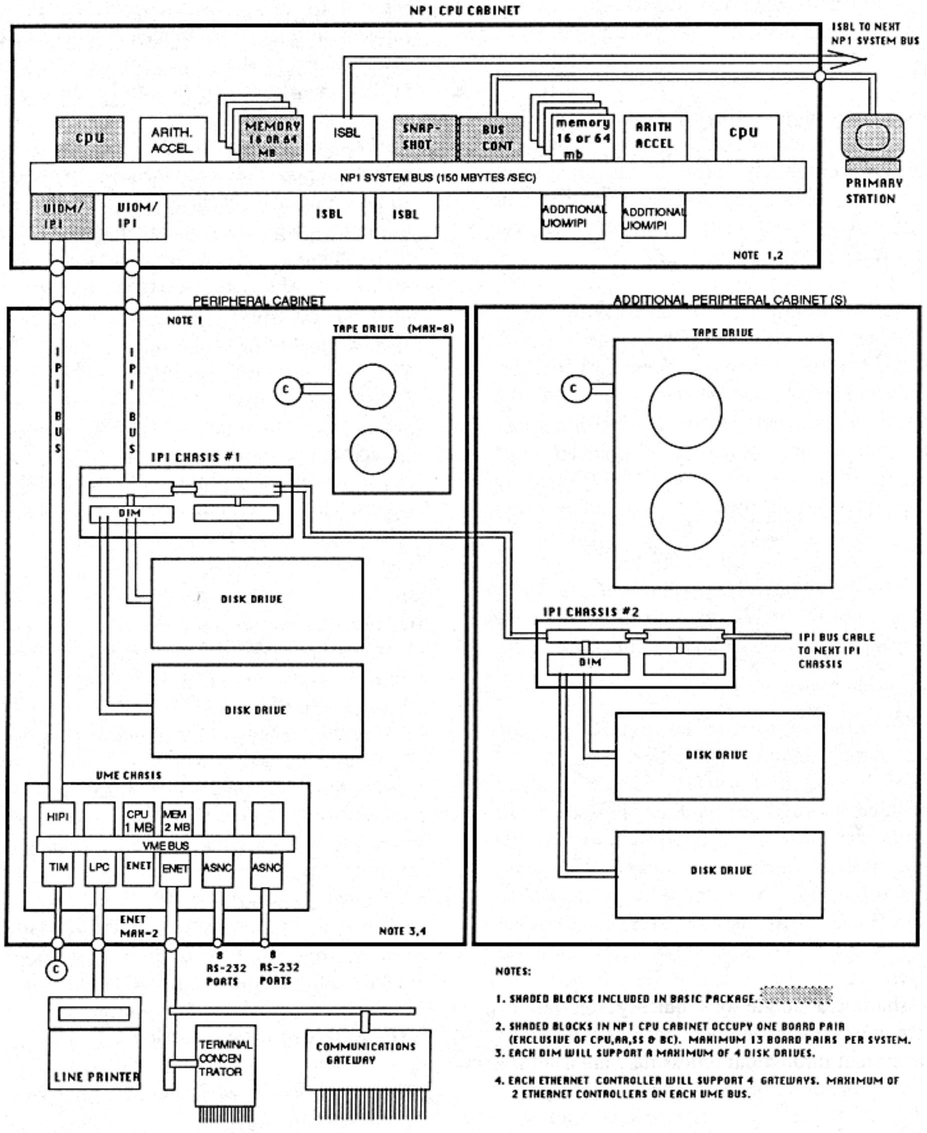

Figure 1. A Typical NP1 System Organization.

**farm** or **processor cube**. As discussed in [13], a **farm** comprises a small number of (mini, supermini, minisuper, or super) processors, typically less than eight, that are communicated through some form of shared memory which they access over a bus or a network of direct processor-to-processor connections. The processors are normally autonomous and can be used as an independent stand-alone computer. It may run existing software in a parallel mode with minimal modification. In this case, a parallel compiler is provided which can recognize the parallelism in existing code (especially Fortran) in order to efficiently distribute parallelized tasks to amongst the available processors.

A **cube** normally includes a large number of processors - generally a power of two, perhaps 64, 128, 1024, or 65,536 (massively parallel). Most cubes are organized from processors based on single microprocessor chips, with additional memory and often special-purpose arithmetic units. Some cubes may have multiple processors on a single custom chip. Each processor has its own local memory and they communicate over a network which is often topologically equivalent to the edges of a multidimensional cube - a hypercube [14, 19]. The processors in a cube are not autonomous and cannot be used as independent computers, but instead the whole network is driven through a separate host computer, usually a mini or a workstation [14]. The software in general must be specially developed, often starting from a new parallel algorithm to make efficient and simultaneous use of a large number of processors.

Within the context of the above discussion, NP1 can be viewed as a large-grained parallel multiprocessing system organized as an 8-processor farm. In the next section, the rationale behind the interconnection strategy adopted for NP1 will be discussed.

## 4. SYSTEM INTERCONNECTION

In a multiple processor design, whether it is farm or cube, the designer's major concerned issues are performance and cost.

Performance constraints are mainly related to the question of how the network is to be organized to avoid it being a substantial bottleneck on actual processing power? For example, a performance problem in a memory-shared system is in memory interference where processors making references to shared memory (and to the interconnection network that associates the processor elements with memory modules) get in one's another way. This would result in delays at the memory interface for data and instruction reference which in turn add to the overhead of processor-to-processor communications [15].

The cost issue involves the selection of the network topology, the implementation of the chosen solutions, and the interconnection complexity. Interconnection complexity refers to the number of links and switches required to connect the processing elements or buses together, and should have a direct relationship to the real cost of implementation.

**Network Topology.** There are many types of network topologies that were discussed in the literature: grids, trees, hypercubes, binary n-cubes, banyan networks, shuffle exchanges, butterfly networks, etc. Each has its own advantages and disadvantages. However, the most common interconnection strategy is to directly link a unit in a network to every other unit. In this scheme, the designer must decide whether to pay the cost of a full interconnection network in which any processor can establish its own direct communication with any other processor at any time, or choose to accept the communications penalty for connecting all of the processors to a single bus which must have a very large bandwidth to satisfy them all along with I/O processors activity. In other words, should we use multiple bus or single bus to connect all of the processing elements?

The use of bus in multiprocessor is based on the simplicity of the bus structure, its well understood electrical properties, low cost, and readily assembled from existing technology [16]. However, buses are restricted by the decreasing capacity they provide with increasing numbers of processors. A single fast processor may be able to use up the entire capacity of a backplane bus especially if it is fetching instructions and data in a burst mode. In a single bus system, all N processors have to share the same link, so the communications capacity per processor is inversely proportional to the number of processors (1/N). As a result, despite the cost advantages, processors sharing a single bus quickly run out of communications capacity when their numbers increase and thus might severely affect the system throughput unless they are used for applications with low communication requirements.

**Bus Capacity.** One solution to the throughput problem is to use more buses with fewer processors on each and to find an efficient way to interconnect these buses together. This brings up the bus adequacy issue where we have to decide how many processors should be connected to a single bus. One rough measure of the adequacy of the bus interconnection is to compare its throughput with the performance of attached processors. If a processor is running at 1 MIPS it may well generate a traffic of the order of well over 1MB/sec in passing instructions and data to and from shared memory accessed over the bus. It should be noted that for each instruction execution, there is at least an instruction fetch (generally four bytes long), and possibly oprerand fetches and putaways. Allowing for overhead and peaks activity, 1MB/sec of bus capacity per MIPS of processing power should be the very minimum provided in an actively parallel system, and 2 to 4MB/sec would be more of an ideal situation. On the other hand, if the system is intended rather as a multicomputer with a low level of intercommunications, and where each processor has an independent access to its own I/O and other resources, then a lower proportion of bus capacity may be sufficient [13].

**Shared Memory Vs. Cross-bar.** The alternative to a bus for implementing high performance processor farm are either to provide direct access to shared memory or some form of cross-bar network [17]. These allow much higher bandwidth at a much higher cost.

Supercomputer farms like the Cray-2 and Cray X-MP usually provide for direct interconnection by access to shared memory. The Cray-2 provides each of its four processors with direct access through four memory ports to each of 128 memory banks, a total of 512 channels of 16 MB/sec capacity each. Processors each have high-bandwidth direct links to the shared memory, with shared memory registers to provide signalling for high speed synchronization between them. Each processor sees the registers as part of its own hardware so that there is essentially no communications latency. There will be no waiting for messages to arrive. The cost of providing direct memory links goes up with the square of the number of processors. The major disadvantage of the strategy is its inflexibility. The number of processors which can be connected is determined by the link and shared register hardware, and different hardware has to be developed and supplied for configurations with different number of processors.

An alternative form of full interconnection is used by the Alliant FX/8 [20], the cross-bar switch. A cross-bar network provides a logical connection between every one of N outputs and N inputs by using switches at the crossover points between connectors from each output and input. This provides what is called a "non-blocking" network, in that any output can communicate with any input, which is not already in communication with another output, without any risk of being blocked by another call already established between a different input-output pair [13]. Clearly this is more desirable as is the fact that any pair of processors is separated by only one switching stage. But the cost of penalty remains. A crossbar system with M processors, N memory modules, and N buses will cost in the order of O(M*N); assuming a bus is connected to all processors but to only one memory module [15].

## NP1 System Interconnection

The NP1 system is a symmetrical tighly coupled multiprocessor with the folowing features:

- All processors are homogenous, identical in architecture and design.
- All processors have full addressability over the addressing range of the system. The full physical addressing range of the system is 4 gigabytes partitioned into 1 gigabyte per system bus. Since the memory is contiguous, all processors in the system have access to all 4 gigabytes and they all should get service from memory at effectively the same rate.
- There is a single operating system (NP1-Unix) responsible for all resources that are considered "global" across the context of the system, as well as those that are considered "local" within a particular bus.

The tightly coupled of four system buses is needed to achieve the concurrency and close cooperation within the structure of a single process, and memory sharing is the preferred approach for processes and processors synchronization.

In a memory-shared system, there is a property called "space-time coherence" [12]. In a space-time-coherent system, it is possible to represent the status of a complete system at a particular time, and it is also possible to shift resources to processes and processors with great efficiency. The property comes from the shared memory that enables the single operating system to manage and inspect all resources within a context of a single system. This property exists throughout NP1. As an example, any processor may execute the algorithms of the single operating system upon shared system status data and resources. The relationship between NP1 processors is built upon the master and master relationship. As an example, any interrupt in NP1 can be targeted to either local or global, and any processor can service any interrupt.

**Memory Protection.** Another concern in connection with memory sharing is security, the protection between virtual address spaces of operating processes. NP1 provides a hardware implemented virtual memory management scheme which permits full utilization of all available memory. It has the following characteristics:

- The 4 gigabytes virtual address space is divided into 4 MAPed quadrants, each quadrant can address 1 gigabyte.
- Two quadrants are reserved for operating system use while the other two quadrants are for user processes.
- System software is "fenced" from user software.
- Segmentation and paging are used for virtual address translation. There are 256 segments, 2048 pages per segment with 8K bytes per page granularity.
- Read, write, execute protection on a per page basis.

Under NP1, memory is available under two operational environments: un-MAPed or MAPed. In the un-MAPed mode of operation, no logical address to physical address translation is done. All memory references are to physical memory locations. In the MAPed mode, the hardware memory management allows the operating system and the user software programs to be loaded into and executed from any location in physical memory. When the MAPed mode of execution is enabled, dedicated CPU hardware and firmware are used to accomplish the virtual to physical address translations.

NP1 uses the Memory Allocation and Protection (MAP) fences to prevent or limit the user software from accessing the system address space and in certain circumstances, the system software from accessing the user address space. There are three types of fences: privileged instructions, domain fields, and address space. Privileged instructions may be executed only from within the system address space. An attempt to execute a privileged instruction from the user address space will generate a privilege violation trap.

The access domain is a structure which dictates memory read, memory write, and execution privileges of a specific page of memory. The domain fields are defined in the page table entry. An access violation trap is generated when the CPU detects a memory access attempt into an area of memory for which the current operating domain read, write, or execute permission bits are not set correspondingly with the type of access being requested.

The most significant bit of the virtual address determines if the referenced address is within the user or the system address space. The execution of software which resides in the user address space is prevented from branching into the system address space or from accessing an operand which resides in the system address space. User space to system space access is denied regardless of the state of the selected domain bits. An attempt by the user to cross into the system space generates an address fence error trap.

**System Availability.** High-availability systems may or may not share memory. A high availability system, unlike a parallel processors, tends to minimize the interdependency among processors since increased interdependency reduces availability. The probability that all processors will be down is lower in an eight-processor unit than in a two-processor system. However, so does the probability that all processors will be up. However, by having at most two processors per system bus, and using the full interconnection through shared memory, NP1 becomes a highly available system. A failure of a processor element, a system bus, or a memory module will not bring the whole system down. The performance and/or throughput of the system may be degraded but the remaining elements are still up and working.

**Configuration.** Figure 2 shows a full 8-processor NP1 configuration. In this type of connection, all system buses are completely connected. The ISBLs are used as a connection link between each pair of system buses. For a full connection, three ISBLs are needed per system bus; two for the two neighboring buses and one for the opposite bus.

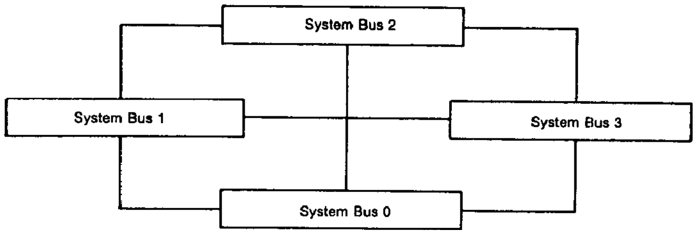

Figure 2 - Full NP1 System Interconnection.

Having established the architectural basis for the system, the remaining sections discuss some details of the NP1 system interconnection, beginning with the system bus followed by specific discussion on the intersystem bus link.

## 5. NP1 SYSTEM BUS

**Organization.** The NP1 system backplane is organized into general bus ports where each port consists of a primary and a secondary slot, for a total of 42 slots. Two of the slots are dedicated to the K-Board and the S-Board, 14 are dedicated to the CPUs and their arithmetic accelerators, and the remaining slots are for memories, UIOMs, and ISBLs.

The main system bus traverses the backplane interconnecting all of the general ports, the S-Board, and appropriate processor interface slot. The bus structure comprises several major fields of multiple lines to provide the message as well as control signals under which the system bus operates.

**Message Types.** The system bus supports three types of message format: (1) Memory Read Transfers, (2) Memory Write Transfers, and (3) Control Transfers which include Data Return Transfers. Each system bus transfer occurs in a three cycle sequence consisting of a polling cycle, a transfer cycle, and a status cycle.

The polling cycle starts when a bus device issues the request for transfer to the K-Board. The K-Board, as the bus priority arbitrator, examines all incoming requests and issues the select signal to the highest priority recognized. When a select signal is received by the requesting bus device, it may make the actual bus transfer in the next cycle. The cycle immediately following the transfer cycle is defined as the status cycle. During this time, the bus device which received the transfer will indicate this by a transfer accepted status on the bus status line. Any parity error condition which may have occurred on the memory address or data buses during the transfer will also be signaled by the parity check circuitry on the S-Board. These signals are available to the transmitting device for capture and test at the close of the trigger clock in the status cycle.

**Major Bus Fields.** The system bus is functionally grouped into seven bus fields: (1) Memory Address Bus, (2) Data Bus, (3) Expanded Data Bus, (3) Source Address Bus, (5) Memory Transfer Code, (6) Operation Code, and (7) Control Signals.

- The Memory Address Bus is a 30-bit word addressing field plus four parity bits used for communicating with the memory system. When used in conjunction with the byte control section of the Memory Transfer Code field, a total of four gigabytes of memory may be addressed.
- The 64-bit Data Bus and the 64-bit Expanded Data Bus are the main transmission paths for data between the memory system and bus devices. A memory read transfer will result in a memory data return transfer of 64 bits (or 128 bits if the Expanded Data Bus is used). A memory write transfer can be either byte, halfword, word, double-word, or quad-word.
- The 11-bit Source Address Bus represents the physical address of the device making the read transfer. This bus is also used by the memory system as the destination address for the corresponding data return.
- The 8-bit Memory Transfer Code is used by all devices to inform the memory system the nature of a bus transfer, read or write.
- The 7-bit Operation Code identifies No-Operation and four classes of transfers: (1) System Control, (2) ISBL Write Status, (3) ISBL Read Status, and (4) Memory Data Returns.

  The system control set of transfers include the targeted and global interrupt requests allowing bus devices to issue interrupt request to different CPUs, and the ISBL clear check transfer used for cache coherence consistency.

  The ISBL write and read status transfers are used for reporting transfer errors between buses while the memory data returns codes are for differentiation of data return types: diagnostic, semaphore, and 64- or 128-bit data.
- The 32-bit control signals are for bus identification, real-time clock, external event, system reset, memory inhibits, etc.

**Bus Naming Convention.** Each system bus in NP1 is assigned a system level bus number at configuration download time by the Control Unit on the K-Board. As each system bus is identified (0 to 3), the K-Board for that bus will drive the bus number assigned on the backplane thereby providing this information to all bus devices which reside on that bus. This bus number is important because it represents a part of a bus device physical address. For memory addressability, the maximum of four gigabytes of memory is linearly partitioned across the four system buses with a maximum allocation of one gigabyte per bus. This allows direct mapping the bus ID number against the two most significant memory addresses bits for determination of the target memory for memory read and write transfers. All other bus devices (CPU and I/O) are identified by the direct mapping of the bus ID number to the two bus number bits in the source and destination address fields.

The system expansion is defined such that all single system bus systems are assigned as system bus 0. When the second bus is connected it is assigned as system bus 1, and the third and the fourth will be known as system bus 2 and 3 respectively.

## 6. INTER-SYSTEM BUS LINK

**Basic Functions.** The ISBL performs two basic functions: (1) to provide inter-system bus communications between functional units on different system buses, and (2) to return status responses if transfers cannot be completed or are in error.

An ISBL, acting as the intercommunication path between system buses, is required to pass through itself most of the transfers issued by other types of bus devices. Without interference, the transfer time through the ISBL from one system bus to the second system bus is one clock (52nsec) period.

The ISBL also has the responsibility of reporting when these pass through transfers encountered, or cause an error condition when the ISBL issues them on the targeted bus. These error reporting transfers are formatted by the ISBL for transmission back to the device which originally issued the transfer that caused the error condition.

**Physical Description.** The ISBL consists of a two board pair, a primary and a secondary board. The primary board handles setup, addressing, control and error detection functions. The secondary board performs 64- or 128-bit memory transfers to and from an identical ISBL board located in a different system bus.

Each ISBL will have a transmit pipeline and a receive pipeline. When two buses are connected via ISBL's, the TRANSMIT section of the first ISBL is connected to the RECEIVE section of the second ISBL, and the TRANSMIT section of the second ISBL is connected to the RECEIVE section of the first ISBL.

Since the interconnection strategy adopted for NP1 is direct, two ISBLs are required for each pair of buses. For a full interconnection of four system buses and eight CPUs, a total of 12 ISBLs is needed.

**ISBL Advance Transfer Notification Control.** The ISBL advance transfer notification control lines provide advance notification to the ISBLs of an impending transfer to them. A set of lines are wire-wrapped between each primary or processor interface slot and the K-Board. Each set of lines consists of two request bus signals and are twisted with a ground wire. The use of two bus request signals allow for flexibility in allowing certain types of transfers to occur on the system bus.

Bus request signals are used in conjunction with the main transmission request/select lines and are directly related to them by priority. After the K-Board has determined the highest priority transmission request, it will examine the matching set of request bus signals. If the two-bit code signal does not match the local bus number, the request bus signal will be decoded to drive one (of three) additional signal wires wrapped in the backplane. These signal wires are connected between the K-Board slot and the three highest priority slots of the backplane. The number of wires placed is system dependent. The dual processor backplanes are capable of supporting the maximum of three ISBLs and will have those lines wired to slot priorities 0, 1, and 2.

**Normal Transfer Operation.** The ISBL, as a bus device, may issue the following transfers onto the system bus in normal operation:

1. Pass through of all memory read transfers.
2. Pass through of all memory write transfers.
3. Pass through of control transfers specified as:
    - Interrupt request, targeted or global
    - Diagnostic data return
    - Semaphore data return
    - 64-bit data return
    - 128-bit data return

The issuance of the control transfer under ISBL read status specified as ISBL clear acknowledge. This transfer is formatted by the remote end of an ISBL in response to a bus device issuing the control transfer ISBL clear check.

The ISBL clear acknowledge is transmitted to the bus device which issued the ISBL clear check.

**Inter-System Bus Communication**

The ISBL was designed to support three types of message formats similarly to those supported by the system bus: (1) Memory Read Transfers, (2) Memory Write Transfers, and (3) Control Transfers.

**Memory Read Transfers.** Memory read transfers require that the memory inhibit be tested if the transfer is targeted to the local bus memory. The appropriate link inhibit is to be tested if the transfer is between system buses.

**Memory Write Transfers.** Memory write transfers targeted across ISBLs require the appropiate link inhibit to be tested. When a system level (cache coherent write) memory write transfer is to be issued to the local memory, the bus device must test all link inhibits along with the appropriate memory inhibit. This system level write must be sent through all ISBLs present in a multiple bus system to invalidate cache memories in devices on the other buses. When this type of transfer is issued, the transmitting device is required to also drive the targeted bus transfer signal on the backplane. This signal indicates to the ISBL that this transfer is to be accepted. As each ISBL accepts the transfer, it must release the common link accept signal to allow the transmitting bus device to determine if all of the ISBLs actually accepted the transfer. If all ISBLs do not accept the transfer, then a system cache error exists and must be reported to the appropriate software element.

**Control Transfers.** Control transfers to or through an ISBL require that the bus link inhibit lines be tested before request polling and transfer.

The specific transfer types passed between the system buses are:

- Memory read of 64 and 128 bits
- Memory reads of 64 bits uncorrected
- Reads of memory error log
- Memory writes of 32, 64, 96, and 128 bits
- Cache coherent writes
- Set and zero semaphores
- Set and zero bits
- Memory data returns of 64, and 128 bits
- Interrupt requests to CPUs

**Status Response**

When the remote end of the ISBL issues transfers onto the system bus, errors may occur which must be reported to the originating bus device. The ISBL status transfers will be issued back to the device which issued the initial transfer only if the error condition occurred on the remote system bus. If the error is initially detected at the local bus, the transmitting device will already have been notified of the error condition occurring and the ISBL will not respond with any error transfer. The general types of responses for the read, write, and control transfers status are:

1. **ISBL Read Status.** The ISBL read status set of transfers are primarily used to respond to errors on transfers which the transmitting device would normally expect responses to. These error transfers mainly replace an expected data return transfer to the original transmitting device, as long as the error did not originate on the local bus of the transmitting device when the original transfer was issued. Thus, the ISBL has to monitor the error condition on the originating bus in order to determine if the ISBL Read Status transfer are to be issued. The ISBL may issue the following ISBL read status transfers:
    - Non-present device
    - Data parity error on semaphore transfer
    - Address parity error on the memory read or semaphore transfer
    - Address and data parity error on semaphore transfer
    - ISBL clear acknowledge
2. **ISBL Write Status.** When a memory write transfer is issued to the memory system on a remote bus and the local status checks have been successful, the bus which originated the transfer may receive a transfer from the ISBL if errors were detected on the remote bus when it was issued by the ISBL. This transfer is called an ISBL write status transfer and it is used to report: (1) Non-present device, (2) Address, data, or address and data parity errors, (3) Non-present device on request interrupt transfer, and (4) Cache coherent memory write broadcast error.

    The cache coherence broadcast error occurs when the remote end of the ISBL is unsuccessful in retransmitting a cache-coherent write back through one or more of the other possible ISBLs on the remote bus. If no other errors occur, the addressed memory on the remote bus is modified. Any failure to rebroadcast the cache-coherent write transfer could leave a cache-based device with invalid data contents. The cache broadcast error must be reported to the appropriate software element by the device which originated the cache-coherent memory write transfer.

## Cache Coherency Support

The cache memories contained in each NP1 processor provide local high-speed access for that processor to information which is a duplicate of that contained in the main memory system. With the main memory being distributed across a maximum of four system buses, a cache memory may contain copies of main memory data from any memory system across the multiple bus system. When the contents of any portion of memory are modified, it is important that this be conveyed to all bus devices in a multiple bus system. Otherwise, inconsistencies may occur between cache and memory, and also between different caches.

The enforcement of multicache consistency in NP1 is to tag memory transfers as cache coherent or non-cache coherent, and to use the write through policy in all memory update. Since the only transfers which modify memory contents in the NP1 system are memory write transfers then under normal conditions, every memory write transfer that is issued would have to be transmitted to every system bus. This effectively causes a maximum of three additional transfers to be carried through the ISBLs. In order to minimize these extra transfers, some of the memory write transfers are divided into bus level (non-cache-coherent) and system level (cache-coherent).

Any bus device which is issuing memory write transfers to a sequentially addressed buffer is required to issue the system level write transfer on the last transfer issued, or at the last address of the cache block. The rest of the transfers are to be issued as bus level writes. This implementation reduces the number of memory writes which must be transmitted to all buses to only one write out of a block of eight double-word writes issued (or one out of 16 single-word writes). Memory write transfers issued to a buffer, where the buffer is not sequentially addressed, must always issue system level transfer.

A cache-coherent write may be transferred to the memory on another system bus or to local memory. If a cache-coherent write is transferred to the memory on another system bus, the transmitting device is only required to check for the transferred condition of the status line. However, a cahe-coherent write to local memory requires that the transmitting device insure that the transfer is accepted by the ISBL. This is accomplished by testing the state of the common link accept backplane signal. This signal is the ored accept condition from the ISBL(s). The transmitting device tests the state of this signal at the end of the status cycle. If the signal is tested in the low condition, all ISBLs have accepted the transfer and no error condition exists. If, however, this signal is high when tested, one or more of the ISBLs will not have processed the transfer and an error must be generated by the transmitting device to notify the system software that a cache memory level system error was detected. The transfer through the ISBLs invalidate the cache memory entries in devices on the remote system buses. If the transfer is not processed through an ISBL, a cache memory in a device on a remote bus may contain data which is not a true copy of memory since the transmitting device modified the memory on its local bus. If the transmitting device detects that either an address or data parity error condition occurred, the cache error is not reported since the meory aborts processing of the write transfer and real memory contents are not modified.

## Shared Process and Data Support

In NP1, the protection of shared process or data areas in memory is accomplished by the use of access control bits in memory called semaphores. In a multiple bus configuration, semaphores can be modified by any processor on any system bus. As a result, if the intended operation fails to go through the ISBL, system deadlock might occur. To ensure all cache-coherent memory write transfers have in fact successfully passed through the ISBL, the ISBL Clear Check bus transfer must be issued by a bus device prior to modifying a semaphore. Thus, a write transfer that may affect other memories in the system is guaranteed to complete. The implementation of the ISBL clear check is as follows.

When a processor gains access rights to a shared area (by a successful set semaphore transfer), it must issue the ISBL clear check transfer to all ISBLs in the multiple bus network and wait until an ISBL clear acknowkledge is received for each ISBL clear check issued. It must then clear any cache-coherent memory writes which may have been temporarily staged in the ISBL internal registers. It may now begin to use the shared area.

Non-cache based memory devices may begin use of the shared area as soon as the access rights are granted.

When any bus device prepares to release its access rights, it must issue the ISBL clear check to all ISBLs and wait for all ISBL clear acknowledge transfers. Once this is complete, the device may release its access rights by a successful zero semaphore transfer.

## Performance

The ISBL is a pipelined structure which operates at the same clock frequency as the system bus (at 52nsec). The transfer rate for the 64-bit memory transfers is 154MB/sec. With the addition of the 128-bit transfer, the transfer rate is increased to 308MB/sec.

## Diagnostic Features

For a single bus system, two modes will be used to establish a transmit and receive path using the resident ISBL board pair. They are:

- Internal loop-back mode
- External loop-back mode

The internal loop-back is included in the ISBL hardware and is strictly used for diagnostic purposes. This path is controlled by one bit in a soft-loadable mode register and it enables a direct path from the transmit portion of the logic to the receive portion of the logic on the same board. Thus, the entire input and output path of one ISBL, including all storage elements, can be functionally tested.

The external loop back mode is used to verify the ISBL data path one step beyond internal loop back mode. This path is also selected by one bit of the mode register. In this mode, the transmit connector will be connected to the receive connector on the same board via a cable. This cable will be the required length to connect two ISBLs on separate buses. Therefore, it will allow for checking of the transmit path out through the transmit connectors onto the cable and back into the receive connectors and the receive logic. It will also test timing of transfers over the cable length. External loop-back simulates the real link to link hook-up between two ISBLs.

## ISBL Serial Link Gate Array

The ISBL is implemented in hardware and there is no firmware involved. The control of ISBL is done via the control unit on the K-Board through the SLGA. The SLGA communicates with the

ISBL through the following commands:

- Load mode register - Load the mode register with the 16 bits of the data field.
- Read mode register - Reads the contents of the mode register plus two bits that provide the online signals of both the local and remote boards.
- Load visibility register - Loads the visibility register with the 16 bits of the data field.
- Read visibility register - Reads the contents of the visibility register.
- Load shift register - Loads the SLGA and the control logic shift register with the data at the input. The visibility register must be set up with the desired conditions before issuing this command.
- Read shift register in block A and shift - Reads the output of the shift registers in block A, and causes the shift register to shift.
- Read shift register in Block B and shift - Reads the output of the shift register in block B, and causes the shift register to shift.

## 7. CONCLUSION

This paper discussed some of many issues involved the design and implementation of the NP1 system interconnection through the inter-system bus link. The primary goal of the ISBL is to provide a bi-directional communication path between different system buses in the most efficient and fastest way. With the capability to issue transfers on every bus cycle and a minimum bandwidth of 154MB/sec, the ISBL has achieved its intended purpose.

## 8. ACKNOWLEDGMENTS

The authors are indebted to Bill Hamilon and George Dragner for their endless effort in debugging and making the system becomes successfully operational.

## References

[1] Alexandridis, N.A., *"Architectural adaptation in image processing supersystems,"* Proc of the 1rst Int Conf on Supercomputing Systems, 1985, pp. 173-181.

[2] Gait, J., *"Two tier scheduling in partitioned multiprocessors,"* Proc of the 2nd Int Conf on Supercomputing, 1987, pp. 179-186.

[3] Gaudiot, Jean-Luc et al., *"The TX-16: A highly programmable multi-microprocessor archiecture,"* IEEE Micro, Oct. 1986, pp. 18-31.

[4] Matelan, N., *"The Flex/32 multicomputer,"* Proc of the 12th Int Sym on Comp Arch, 1985, pp. 209-213.

[5] Gajski D. D. and J. Peir, *"Essential issues in multiprocessor systems,"* IEEE Computer, June 1985, pp. 9-27.

[6] Kasahara, H. and S. Narita, *"An approach to supercomputing using multiprocessor scheduling algorithms,"* Proc of the 1rst Int Conf on Supercomputing Systems, 1985, pp. 139-148.

[7] Herzog, V., *"Performance modelling and evaluation for concurrent computer architectures,"* Proc of the NATO Adv Res Workshop on Hi-Speed Computation, 1983.

[8] Patel J. H, *"Performance of processor-memory interconnections for multiprocessors,"* IEEE Trans on Computers, Oct. 1981, pp. 771-780.

[9] Vianney, D., *"The Gould NP1 system architecture,"* Proc of the 2nd Int Conf on Supercomputing, 1987, pp. 35-43.

[10] Vianney D. and S. Heffner, *"An overview of the Gould NP1 parallel multicomputer system,"* Proc of the 2nd Int Conf on Supercomputing, 1987, pp. 29-34.

[11] Ngo, J., *"The arithmetic accelerator for the NP1 CPU,"* Gould Inc., Computer Systems Division, 1987.

[12] Lorin, H., *"Systems architecture in transition - An overview,"* IBM Sys Journal, Vol. 25, Nos. 3/4, 1986, pp. 256-273.

[13] Johnson T., and T. Durham, *"Parallel processing: The challenge of new computer architectures,"* Ovum Inc., Princeton, NJ, 1986.

[14] Hayes, J.P. et al., *"A microprocessor-based hypercube supercomputer,"* IEEE Micro, Oct. 1986, pp. 6-17.

[15] Das C.R., and L.N. Bhuyan, *"Computation availability of multiple-bus multiprocessors,"* U of Southwestern Louisiana, 1985.

[16] Gustavon, D., *"Computer buses - A tutorial,"* IEEE Micro, August 1984, pp. 7-22.

[17] Wulf W.A. and C.G. Bell, *"C.mmp-A multiminiprocessor,"* Proc AFIPS FJCC, Dec. 1972, pp. 765-777.

[18] Thakkar S. et al., *"Balance: A shared memoy multiprocessor systems,"* Proc of the 2nd Int Conf on Supercomputing, 1987, pp. 93-101.

[19] Fox, G.C., *"The hypercube as a supercomputer,"* Proc. of the 2nd Int Conf on Supercomputing, 1987, pp. 186-194.

[20] *"FX/Series product summary,"* Alliant Computer Systems Corp., 1987.