Extended Abstract For
IEEE Multi-Chip Module Conference
MCMC-96

# SPACE-CUBE: A FLEXIBLE COMPUTER ARCHITECTURE BASED ON STACKED MODULES

July 18, 1995

Gary Bolotin
bolotin@telerobotics.jpl.nasa.gov

Jet Propulsion Laboratory
California Institute of Technology
4800 Oak Grove Drive
Pasadena, CA 91109

## 1. Introduction

Future computing systems need low volume, mass, and power electronics. Through the use of innovative architectural, and microelectronic packaging design this goal can be achieved. These architectures need to:

1). easily incorporate a variety of different requirements.
2). provide a framework by which data bus protocol and definition, module package size, interconnection and shape, can be standardized in order to maintain modularity and architectural flexibility.
3). create an environment that encourages module inheritance from mission to mission.
4). be independent of processor type.
5). incorporate new technologies as they become developed.

The Space-Cube Architecture addresses these issues.

### 1.1 Comparison With State of the Art

Prior to the Space-Cube, multiprocessor and distributed systems were made up of boards or modules plugged, side by side, in a card cage or frame. The modules were then connected by means of a single backplane or bus along the rear surface of the modules. Unfortunately, this single bus strategy creates a system bottle neck. If multiple busses were required they could be created, but the lack of an underlying framework for multiple bus construction, resulting in the possibility of a different design for every application. The Space-Cube provides this necessary framework.

Backplane interconnections make use of only one surface of the structure. Multiple bus architectures implemented in a backplane would require either crossing busses or nonidentical modules. Crossing buses limit the scalability of a design, while nonidentical modules increase the cost of a. design. Space-Cube architecture implementations, by making use of all the surfaces of the modules, are implemented with noncrossing, module to module connections, and identical module construction. This eliminates the scalability restriction and reduces cost at the same time.

## 2. Space-Cube Architecture

The Space-Cube architecture is a flexible, processor independent, computer architecture based on stacked interchangeable modules. 'The modules are regular polygon (i.e. square, hexagon, etc.) in shape. These modules when stacked on top of each other, are easily configured to a variety of more complicated architectures. The resulting system is easily expanded and maintained.

## 2.1 The Module Building Block

The basic building block of this architecture is called a module, as illustrated in figure 1. The module shown is in the shape of a square, and subsequently can be placed in any of four possible orientations. (N, S, E, W) The communication between this module and other is done through a high bandwidth bus, represented by the arrow. A single bus is confined to fit within one side of a module. A module that interfaces to more then one bus would have more then one arrow.

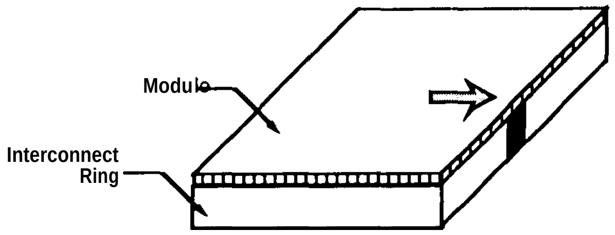


Figure 1. The basic module

## 2.1 **Interconnect Rings**

Interconnections between modules are done by means of interconnect rings, as shown in figure 2. An interconnect ring is a hollow four sided structure which can carry up to four buses from module to module. Any side of the interconnect ring can (not) independently conduct the bus to its neighboring module depending on whether it is conductive or nonconductive. This is illustrated by a thick black line or lack there of. These rings, just like the modules, can be placed in any of four possible orientations. (N, S, E, W) The interconnect ring is achieved by using connectors which provide the electrical, thermal, and mechanical interface between the modules of the system.

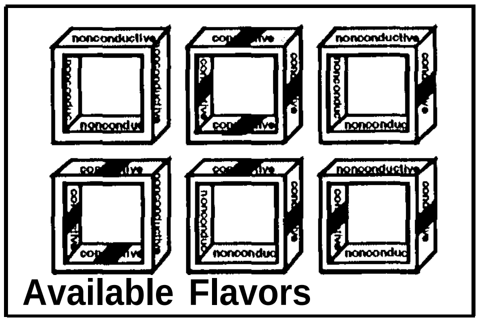


Figure 2. Interconnect Rings

A module stack is constructed by stacking modules together, leggo-style, with interconnect rings as shown in Figure 3. This results in a physically and mechanically robust system.

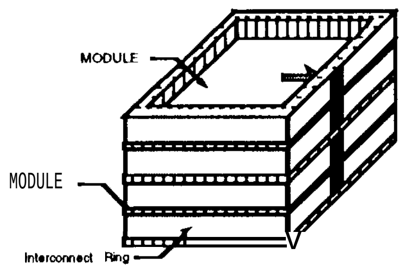


Figure 3. Module Stack

A structure of multiple stacks is constructed by tiling module stacks together to form a 3-D structure as shown in Figure 4.

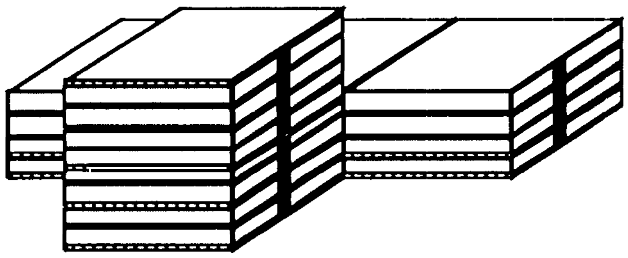


Figure 4. Structure of Multiple Stacks

## 2.2A Space-Cube Based Computer System

To illustratehow the space-cube architecture can be used to construct a computer system, we first need to construct several types, “flavors” of modules. The granularity of the modules is to be chosen based on the problem and technology at hand. The granularity must be small enough so that there is sufficient commonality among systems to justify this approach. The granularity must also be chosen large enough to overcome the overhead of this bus based architecture.

## 2.3 Module Flavors

For purposes of this proposal we will discuss four possible module flavors:

1). CPU

2). Memory

3). Gateway

4). 1/0

These modules types will be discussed in detail and are illustrated in figure 5. Systems can then be created by linking the desired modules into a stacked design.

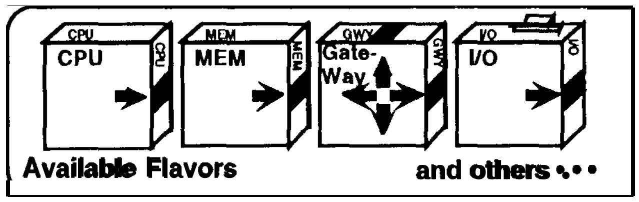


Figure5. Module Flavors

The CPU flavor is a self contained processing module. This module contains a microprocessor along with its local bus and a single interface on to the Space-Cube bus: Space-bus. The CPU will access the Space-bus when it is performing a read or write of external data. The remainder of the time the Space-bus is free to be used by other modules.

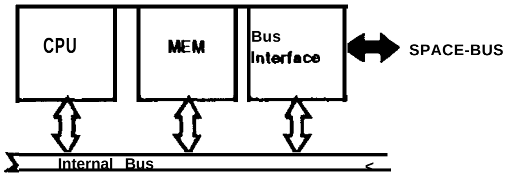


Figure 6. CPU

The memory flavor is a single ported memory unit mapped onto the Space-Cube bus. This unit represents the mass storage of a Space-Cube stack. The memory module functions as a slave module and will only respond when accessed by the current bus master on the Space-bus.

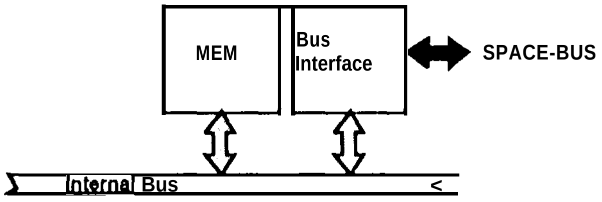


Figure 7. Memory

The I/O flavor is a single ported unit and presents a means of getting data into and out of Space-Cube stack. The I/O module functions as a slave module and will only respond when accessed. The 1/0 module has two external connections. The first is the connection to the Space-bus and is represented by a bus vector. The second connection is from the I/O module to the external enviorment. This connection is represented by an I/O vector.

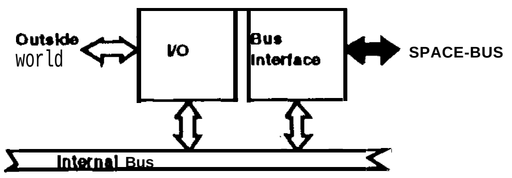


Figure 8. I/O

The gate-way flavor is a four-ported unit and is the only module that is unique to the Space-Cube architecture. The gate-way provides a means of connecting one bus surface onto any one of the three other surfaces. The gate-way flavor is illustrated in the following figure. The gate-way shown is a four ported gate-way and thus requires only two internal busses. For example, if port A is talking to port D, then the only possible remaining connection is from port B to port C. If a request were to come in on a port for a port that is occupied the request will be denied.

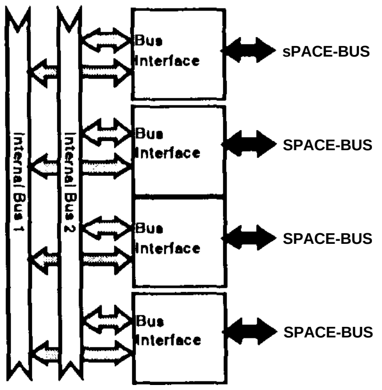


Figure 9. The Gate-way

## 2.4 **Examples**

Module stacks are constructed by simply stacking modules together with interconnect rings. This section describes examples of some Space-Cube systems.

A simple example of a Space-Cube is a simple single processor with its own memory and 1/0. A single processor is constructed by stacking a CPU module together with a memory and I/O module as shown. Since the three modules are all stacked in the same direction, the three modules share the same bus.

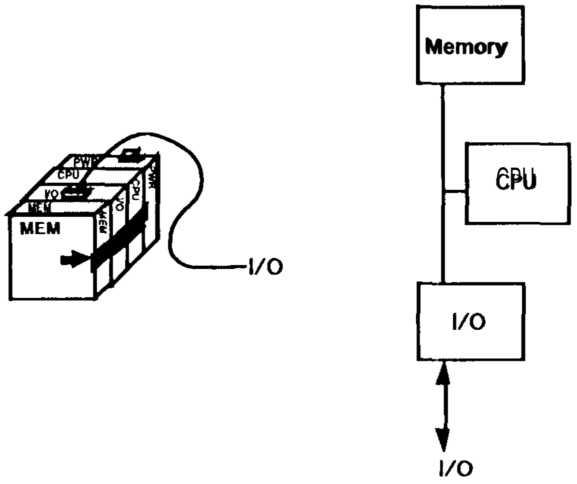


Figure 10. A Single Processor

A simple dual processor, with a common shared dual-ported memory can be constructed by putting the processors, and the shared memory module, at three unique angles, together with a gate-way. The local bus for each processor is assigned a unique surface. The shared memory bus is given a third. The gate-way module which is placed between the processors and the shared memory completes the stack.

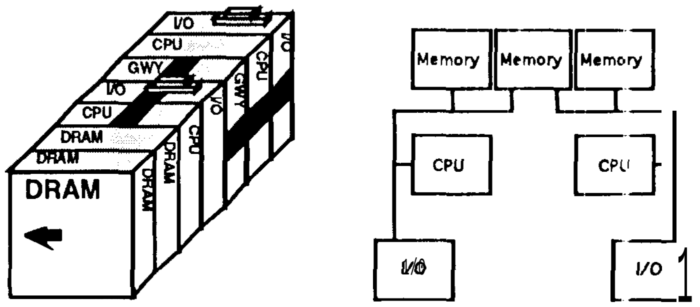


Figure 11. A Dual Processor With Shared Memory

A multiprocessor can be constructed by stacking four processors in each of the four directions, together with a gate-way module. The example shown below is of a seven processor cluster. It is constructed out of seven processor modules, a memory module, and an eight ported gate-way.

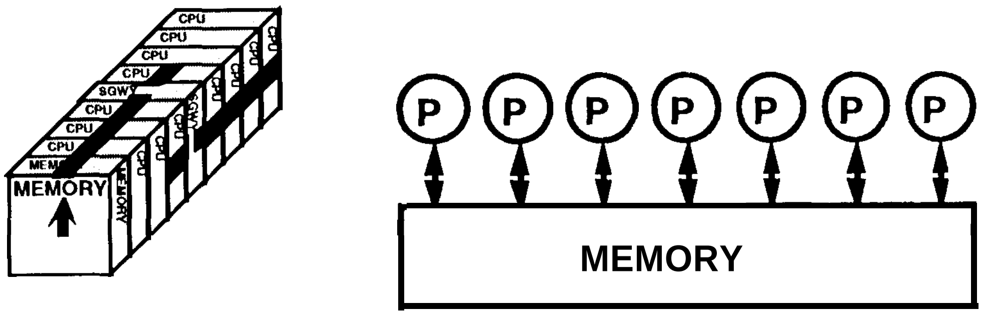


Figure 12. Processor Cluster

Figure 13 is an architecture which was chosen because it illustrates some of the key advantages of the Space-Cube. The architecture shown, under fault-free conditions, results in a distributed processing environment with one processor dedicated to science data processing, and the other to engineering data processing. However, upon the detection of a fault in either processor, the surviving processor will assume the responsibilities of the other. Fault tolerance is made possible because of the cross-strapping between the processors, memories and 1/0s.

The mass and power of this architecture is comparable to a single string option yet it can provide similar fault protection compared to a true dual string architecture. Only the critical I/O hardware is duplicated.

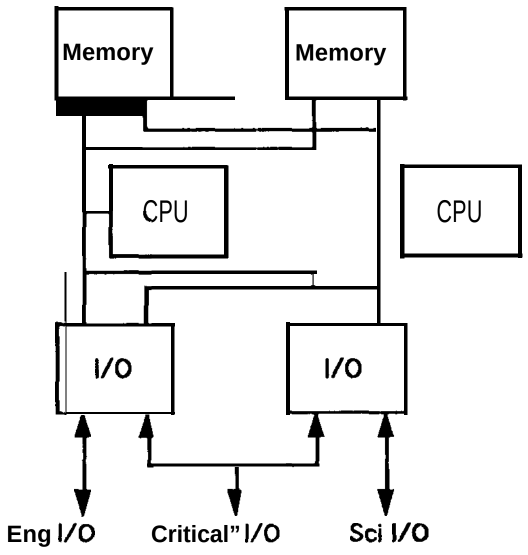


**Figure 13. Space-Cube Demonstration**

This architecture can easily be implemented in a Space-Cube. The following module flavors are needed:

1). Processor
2). Memory
3). Gateway
4). Fault Detection (this module could be combined with the Gateway module but is separated for purposes of illustration)
5). I/O

Figure 13 shows a Space-Cube implementation of this architecture. Each processor is given a side of the cube for its local bus, and another side for its shared bus. The sides of the cube are interconnected by means of the Gateway module. The functionality of the gate-way module will be closely controlled by the fault detection unit module: FDU. The health of the two processing modules will be continuously monitor by the FDU. This module upon detection of a faulty processor will open up the memory and I/O functions to the functional processor.

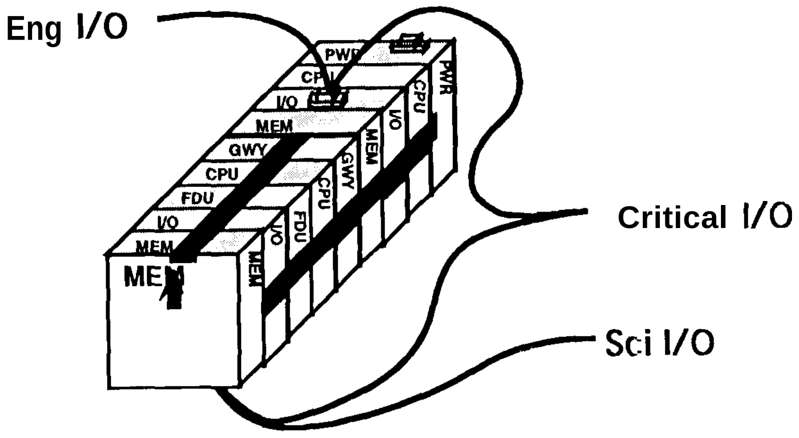


Figure 13. Space-Cube Implementation

## 4. Conclusion

We propose to deliver a standard module framework by which data bus protocol and definition, module package size, interconnection and shape, can be standardized in order to maintain modularity and architectural flexibility. We also will demonstrate the architecture using an implementation that illustrates the key features of the Space-Cube architecture. This implementation will be constructed, using existing modules as much as possible.

This proposal ties together modules being developed by industry sources, together with the dense packaging MCM to MCM technology and Die to Die stacking, and others all under a unifying framework.

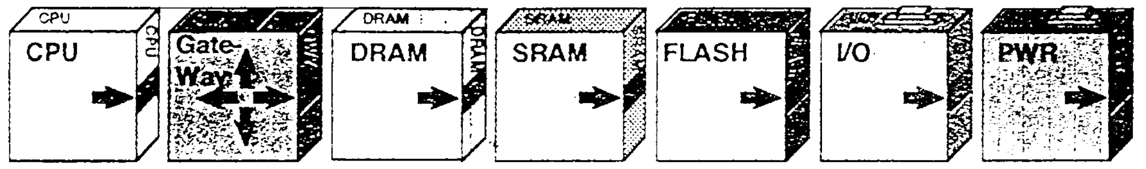
## 7. References

[1] Savio Chau, An Optimal Architecture of Pluto Fast Flyby, Jet Propulsion Laboratory Internal Document, October 16, 1994.

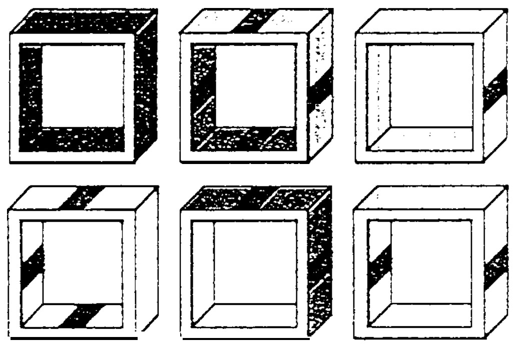
[2] Leon Alkalaj, The Design and Implementation of NASA's Advanced Flight Computing Module, IEEE Multichip Module Conference, Santa Cruz, CA, January 31-February 2, 1995.

[3] Military and Defense Electronics, ISC Develops Mix and Match 3-D Package For AF, Vol. 5, No, 5 April 1994.

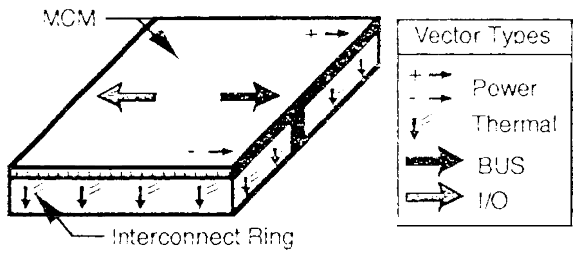
# SPACE-CUBE aRCHITECTURE



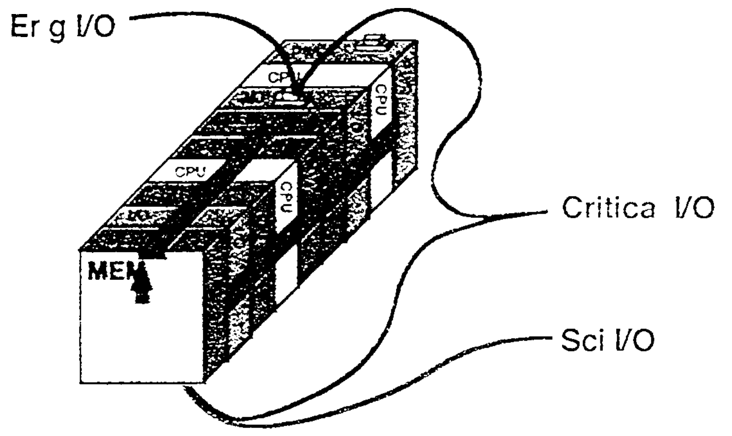
Available Flavors and others • • •

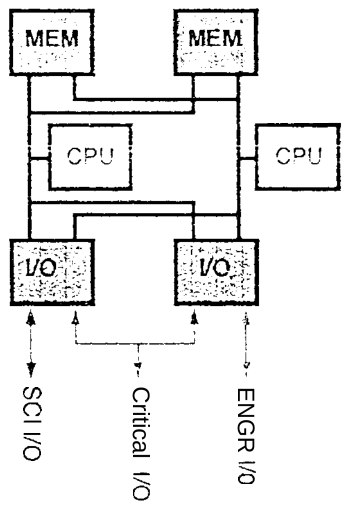


Interconnect Rings



Module Abstraction





Spacecraft Example
(Pluto Express Architecture)

FEATURES

- Processor Independent
- Interchangeable Modules
- Maintainable & Upgradeable
- Standard Features
  - Module Size & Shape
  - Interconnect Bus
  - Interconnect Method
- Can Be Easily Configured To The Following Architectures
  - Single String
  - Dual String
  - TMR
  - Multiprocessors
  - Distributed
  - NonDistributed
  - Others
- Module Inheritance from AFC
- Modules & Interconnections Are Simplified yet complicated architectures can be created.
- Architecture applies regardless of granularity
  - Board Level
  - Stacked MCMs
  - Stacked Dies
- Interconnections do not cross