Space-Cube: A Flexible Computer Architecture Based On Stacked Modules

Gary Bolotin
Jet Propulsion Laboratory
California Institute of Technology
4800 Oak Grove Drive
Pasadena, CA 91109
bolotin@telerobotics.jpl.nasa.gov

Abstract
This paper presents a flexible, computer architecture based on stacked interchangeable modules. The architecture is ideally suited to implementation using stacked multichip modules (MCMs). The architecture, by making use of all sides of a module stack, allows simple, single bussed modules, together with a gate-way module, to be easily configured into a variety of more complicated architectures.

Introduction
Future computing systems need low volume, mass, and power electronics.[1] 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.

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 bottleneck. Multiple busses, when required, could be created, but the lack of an underlying framework for multiple bus construction, results in the possibility of a different design for every application. The Space-Cube provides this necessary framework.

Backplane interconnections, as shown in Figure 1, make use of only one surface of a stack of boards. Multiple bus architectures implemented in a backplane would require either crossing busses or nonidentical modules. Crossing busses 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.

Figure 1. Backplane Technology

Stacked MCM Technology
The Space-cube architecture is ideally suited to implementation using stacked multichip modules (MCMs). Figure 1 shows a 3-D MCM stack implemented using elastomeric elements for the vertical interconnections. This MCM stacking technique is suitable for a spaceborne computer, and is being proposed or used in flight computers of Space Computer Corporation, NASA Jet Propulsion Laboratory and others.[1,2,3]

This module stacking technique replaces the traditional backplane with a 3-D interconnect. In this case an elastomeric interconnect. Elastomeric connections are

U.S. Government Work Not Protected by U.S. Copyright
made out of a compressible material that conducts only in the vertical direction. The stack shown is held together by 4 fasteners, one in each corner. These fasteners hold the 3-D interconnect elements, under compression, against pads along the perimeter of each substrate. The elastomeric connection allows for separation of the individual elements of the stack for easy repair, replacement or reconfiguration.

**The Space-Cube Architecture**

The Space-Cube architecture provides a framework that makes innovative use of the compact regular features of the chassis-less MCM stack. This framework allows one to efficiently build systems which are difficult or impossible with chassis based designs.

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.

**The Module Building Block**

The basic building block of this architecture is called a module, as illustrated in figure 3. This module represents an abstraction of a single MCM.

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 modules is accomplished through a high bandwidth bus, represented by the arrow/vector. 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.

Even though a module may only require a single bus, Space-Cube modules are required to have the I/O pads of remaining sides connected from one surface to another.

**Vectors**

To simplify the discussion vectors are used to represent properties of modules. Five vector types will be discussed; bus vector, power vector, thermal vector, testability vector, and I/O vector. These vectors represent independent features of a module and can be directed independently of each other. However, for purposes of this paper only the communication bus and power distribution bus are to be discussed. They are to be combined into one vector represented by a heavy black arrow.

The bus vector represents the high bandwidth bus used for basic communication between modules. A single vector represents a unique bus. Modules that need to
communicate with more than one bus will have more than one vector.

The power vector represents the direction by which a module receives its power. This vector may or may not be independent of the bus vector.

The I/O vector represents the direction by which a module communicates with the outside world. Bus vectors and I/O vectors can not point in the same direction if such I/O connections would interfere with module to module communication bus connections on that side of the module.

The thermal vector represents the dominant direction of heat conduction. The thermal vector can be used for analyzing the thermal properties of a stack.

The testability vector represents the direction in which a module would receive the five I.E.E.E. 1149.1 module scan test signals. The use of these signals would allow in system test and debugging.

Interconnect Rings

Interconnections between modules are done by means of interconnect rings, as shown in figure 4. The interconnect ring represents an abstraction of the elastomeric connections shown in figure 2. The interconnect ring provides the electrical, thermal, and mechanical interface between the modules of the system. 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 independently interconnect the bus to its neighboring module depending on whether it is conductive or nonconductive. This physical layer could be achieved by a conductive or nonconductive elastomeric. This connection is abstracted by a thick black line or lack there of. Figure 3 shows all possible ring flavors in a four sided system. These rings, just like the modules, can be placed in any of four possible orientations. (N, S, E, W)

A module stack is constructed by stacking modules together, Lego-style, with interconnect rings as shown in Figure 5. This results in a physically and mechanically robust system.

Figure 4. Interconnect Rings

Figure 5. Module Stack

A Space-Cube Based Computer System

To illustrate how 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, and 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.

Module Flavors

For purposes of this discussion we will discuss four possible module flavors:
1). CPU
2). Memory
3). Gateway
4). I/O

These module types will be discussed in detail and are illustrated in figure 6. Systems can be created by linking the desired modules into a stacked design.

Figure 6. Module Flavors

As shown in Figure 7, CPU flavor is a self contained processing module. This module contains a microprocessor along with its local bus and a single interface with the Space-Cube bus: Space-bus. The CPU will access the Space-bus when it is performing a read or write of external data. During the remainder of the time the Space-bus is free to be used by other modules.

Figure 7. CPU Flavor
The memory flavor, shown in Figure 8, 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.

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 I/O 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 environment. This connection is represented by an I/O vector. See Figure 9.

The gateway flavor is a multi-ported unit and is the only module that is unique to the Space-Cube architecture. The module is basically a four ported crossbar network together with the arbitration logic necessary to make the module work. The gateway provides a means of connecting one bus surface to any one of the three other surfaces. In a four sided system, gateways come in two varieties, four and eight ported. The four ported gateway treats each side of the stack as a single unique bus. A four ported gateway module when inserted into a stack would not represent a break in the bus structure. The eight ported variety separates buses above the module from those below. This results in eight unique buses.

The four ported gateway flavor is illustrated in figure 10. Since only two concurrent connections are possible, this module 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.

The eight ported gateway flavor is illustrated in figure 11. The eight ported gateway, requires only four internal busses. This variety of gateway acts as a separator of the busses above and below the module.
**Single Processor**

A simple example of a Space-Cube is a simple single processor with its own memory and I/O. 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.

![Diagram of Single Processor](image)

Figure 12. A Single Processor

**Dual 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.

This example implementation was chosen because it illustrates some of the key advantages of the Space-Cube. Under fault-free conditions, the system 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 I/Os.

The mass and power of this example 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.[4]

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

1. Processor
2. Memory
3. Gateway
4. I/O

![Diagram of Dual Processor With Shared Memory](image)

Figure 13. A Dual Processor With Shared Memory

The gate-way module provides the means for communication between surfaces of the stack, thus making complicated architectures possible. Figures 14 and 15 show the gate-way in action. Since each processor is assigned its own unique surface, or bus, communication between a processor and its own local memory can be achieved with out interfering with the other processor. A potential conflict can occur only when two or more processors try to access the same shared bus.

![Diagram of Gate-way In Action](image)

Figure 14. The Gate-way In Action
A multiprocessor can be constructed by stacking four processors in each of the four directions, both above and below a gate-way module. Each processor has its own unique bus. Communication between the processors of a cluster is done by means of the gate-way module.

**Conclusion**

The Space-Cube architecture is computer architecture based on 3-D stacked modules. The architecture allows simple single bussed modules to form a wide variety of complicated architectures, by making use of all sides of the module stack. Module inheritance, is also encouraged.

The Space-Cube architecture is not a computer architecture in and of itself, but it provides a means of mapping existing computer architectures into a 3-D stack of modules in an elegant fashion. All connections are done with simple module to module connections, and all without any crossing wires.

Work is underway to demonstrate the architecture using commercial off the shelf components (COTS), PC104 modules together with PCMCIA cards are being used as basic module building blocks. The PC104 cards are adapted to a Space-Cube module by attaching connectors to all four sides of a PC104 card. The design of a gate-way module, the only module that needs to be designed from scratch has also been started.

Although ideally suited for stacked MCM technology the Space-Cube architecture is technology independent. The architecture can be demonstrated with commercial off the shelf components today, and provide a framework for stacked die implementations in the future.

**Acknowledgment**

This research described in this paper was carried out by the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration.

**References**