**Abstract**

The paper presents a flexible, compiler-generated architecture based on Stacked Modos. A Space-Cube: A Flexible Computer.

Gary Polak

*Department of Computer Science*

*University of Pennsylvania*

*Philadelphia, Pennsylvania 19104*

*October 9, 1979*

**Introduction**

The introduction provides an overview of the research and highlights the key contributions of the paper.

**Background**

This section covers the background information necessary to understand the paper.

**Related Work**

This section discusses previous work that is relevant to the current research.

**Methodology**

The methodology section explains the approach used in the research.

**Results**

The results section presents the findings of the study.

**Conclusion**

The conclusion summarizes the main findings and suggests future directions for research.

**References**

A list of references is provided for further reading.

**Appendices**

Any additional material that supports the research is included in the appendices.
The 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.

The Module Building Block

The basic building block of this architecture is called a module, as illustrated in Figure 1. This module represents an abstraction of a single MCM.

![Figure 1. 3-D MCM Stack](image)

Even though a module may only require a single bus, Space-Cube modules are required to connect the I/O pads of remaining sides through from one surface to another. This allows the communication occurring on the other surfaces to be uninterrupted by this module.

Vectors

To simplify the discussion vectors are used. Vectors can be 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 bus and power 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 I/O connections would interfere with module to module 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/I/E. I/I/I/I 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 3. The interconnect ring represents an abstraction of the interconnects shown in Figure 1. 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 (not) independently conduct the bus to its neighboring module depending on whether it is conductive or nonconductive. This would represent a conductive or nonconductive elastomeric. This connection is abstracted by a thick black line or lack thereof. 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 4. This results in a physically and mechanically robust system.

**Module Flavors**

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

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

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.

**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" or 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.

**Available Flavors**

Figure 5. Module Flavors

The CPU flavor is a self contained processing module. This module contains a microprocessor along with its local bus and a simple 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.

**CPU**

The memory flavor is a single ported memory unit mapped onto the Spac.e-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 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.

If a request were to come in on a port for a port that is occupied the request will be denied.

The eight ported gate-way flavor is illustrated in the following figure. The eight ported gate-way, requires only four internal busses. This variety of gate-way acts as a separator of the busses above and below the module.

Examples

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

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.
The following module follows the previous text.

Diagram 12. A Dual Processor with Shared Memory

Diagram 11. A Single Processor

Diagram 10. A Processor with I/O

Diagram 9. A Processor with Memory
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.

**Figure 14.** The Gate-way In Action Cont.

**Figure 15.** 10x's Sol Cluster

**Conclusion**

The Space-Cube architecture is a computer architecture based on 2-1 stacked modules. The architecture allows simple single busied 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, by 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. 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 of 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**


