# Parallel Processing With the TMS320C40 Parallel Digital Signal Processor

## Application Report

**Yogendra Jain**
**Sonitech International Inc.**

PRINTED WITH
**SOY INK**™

**TEXAS INSTRUMENTS**

Printed on Recycled Paper

**IMPORTANT NOTICE**

Texas Instruments (TI) reserves the right to make changes to its products or to discontinue any semiconductor product or service without notice, and advises its customers to obtain the latest version of relevant information to verify, before placing orders, that the information being relied on is current.

TI warrants performance of its semiconductor products and related software to the specifications applicable at the time of sale in accordance with TI's standard warranty. Testing and other quality control techniques are utilized to the extent TI deems necessary to support this warranty. Specific testing of all parameters of each device is not necessarily performed, except those mandated by government requirements.

Certain applications using semiconductor products may involve potential risks of death, personal injury, or severe property or environmental damage ("Critical Applications").

TI SEMICONDUCTOR PRODUCTS ARE NOT DESIGNED, INTENDED, AUTHORIZED, OR WARRANTED TO BE SUITABLE FOR USE IN LIFE-SUPPORT APPLICATIONS, DEVICES OR SYSTEMS OR OTHER CRITICAL APPLICATIONS.

Inclusion of TI products in such applications is understood to be fully at the risk of the customer. Use of TI products in such applications requires the written approval of an appropriate TI officer. Questions concerning potential risk applications should be directed to TI through a local SC sales office.

In order to minimize risks associated with the customer's applications, adequate design and operating safeguards should be provided by the customer to minimize inherent or procedural hazards.

TI assumes no liability for applications assistance, customer product design, software performance, or infringement of patents or services described herein. Nor does TI warrant or represent that any license, either express or implied, is granted under any patent right, copyright, mask work right, or other intellectual property right of TI covering or relating to any combination, machine, or process in which such semiconductor products or services might be or are used.

# Introduction

This paper examines parallel processing using the Texas Instruments TMS320C40 floating-point processor. It demonstrates popular parallel architecture topologies such as hypercube, mesh, ring, and pyramid with the 'C40 and discusses the tradeoffs and performance of these 'C40-based architectures. This paper is divided into the following sections:

- **Overview**

  Tells why the 'C40 architecture is ideal for parallel processing and describes a VME-based 'C40 board.

- **Parallel Processing Topologies**

  Describes different parallel architectures such as hypercube, pyramid, mesh, ring, and tree. Also discusses designing of massively parallel systems using the principle of reconfigurability.

- **TMS320C40-Based AT/ISA and VME Architecture**

  Discusses the architecture of VME and AT/ISA TMS320C40-based boards that are expandable from two nodes capable of 100 MFLOPS (million floating-point operations per second) to hundreds of nodes.

- **System-Level Design Issues With the TMS320C40 Communication Ports**

  Discusses the 'C40 node reset and offers design suggestions.

- **Matrix Multiplication Application of Parallel Processing**

  Explains matrix multiplication.

- **Parallel Processing Architecture Topologies**

  Describes the hypercube and its properties and mesh topologies.

- **Reconfigurable Massively Parallel Processors**

  Discusses software approaches and links.

- **Benchmarks, Analysis, and Data Composition**

  Explains benchmarking and evaluation of parallel processing systems, as well as algorithm efficiency and data decomposition strategies.

- **Conclusion**

- **Definitions**

- **References**

# Overview

Computational demands continue to outpace readily available technology. In recent years, fiber optics technology has revolutionized the rate at which data can be carried between any two points. The theoretical communication bandwidth offered by fiber optic channels is of the order of $10^{16}$ Mbytes/s. Satellite-generated data must be processed at a rate of $10^{10}$ Hz. Before and during surgery, when a team of surgeons require a 3-D view of human body parts on a TV screen using tomography technology, information must be processed at speeds on the order of $10^{15}$ Hz. Many more applications demand increased processing speeds: speech recognition, spatial and temporal pattern recognition, modeling fusion rectors, oil explorations, astronomy, robotics, and the solutions of large differential equations for numerical simulations of earthquakes and for atomic and nuclear physics with thousands of variables. Next generation communication technology will support and link ISDN, FDDI, and ATM data bases; live surgery across continents; and intricate defense networks. No sequential computer or existing supercomputer can meet processing demands today and certainly not in the future.

The silicon technology barrier has almost been reached; pushing closer may reduce reliability and increase cost. One of the few options for meeting the computation needs is to exploit parallel processing. The concept of parallel processing is illustrated in the simple analogy of a farmer plowing his field. Alone, it will take him several months, by which time the harvest season will be gone, and his efforts will have been in vain. Instead, the farmer brings in his four sons, all able to do the same quality of work at the same speed. Each one of them starts from four different directions—say north, east, west, and south—and the father coordinates the activities. Ideally, the total plowing time is reduced by a factor of four. This is precisely how parallel processing works; each processor is similar in clock speed, memory size, and communication rate, and they divide any task among themselves to speed up execution.

## Need for a Dedicated Parallel Processing System

Workstations such as the Sun SPARC, HP 9000 series, and Digital Alpha offer 50 to 200 MIPS (million instructions per second) processing. However, using a workstation as a high-performance computation engine has several drawbacks:

- Complex operating systems can occupy as much as 50% of a CPU's processing time.
- Sophisticated graphics displays and graphical user interfaces demand extensive CPU processing.
- Real-time applications require an additional software layer, which must coordinate the disk, graphics, and peripheral I/O, host-to-data acquisition, and data-to-host transfers.

These drawbacks can be easily bypassed by integrating the workstation with dedicated parallel processing host-based hardware. Such hardware can accelerate an application by several orders of magnitude over the workstation. For example, a dual TMS320C40-based processor board can accelerate the performance of a '486 processor by a factor of 20 to 40 for signal processing applications.

Computer manufacturers have also realized the limited I/O and processing capability of workstations and are gravitating toward providing such standard host buses as the ISA, EISA, Microchannel, SBus, and Futurebus for application-specific hardware. As shown in Figure 1, a host can have numerous subsystems, including a dedicated accelerator and a data I/O.

**Figure 1.  Architecture for Real-Time Processing**
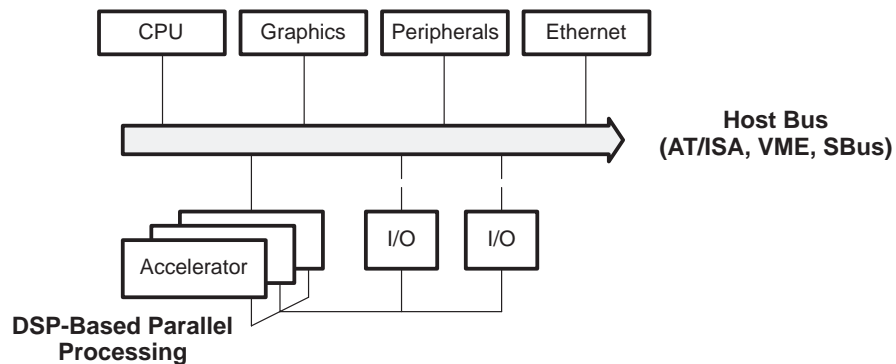


Workstation buses are easily saturated by the various subsystems that interface to the bus. I/O systems with dedicated links to an accelerator can significantly reduce the data transfer burden on the bus as well as off-load the CPU from real-time computation processing.

**Need for a Dedicated I/O Link**

Although these standard host buses are sufficient for nonreal-time data and program transfers and for uploading and downloading of compressed and processed data, they fall short for high-speed data transfers and processing. For example, the ISA bus transfers peak at 2 Mbytes/s, EISA at 5–10 Mbytes/s, Microchannel at 10–20 Mbytes/s, VME at 40 Mbytes/s, and SBus at 40–80 Mbytes/s. An image processing system running at 40 frames per second with color image sizes of $1024 \times 1024$, 32 bits/pixel, and about 100 operations/pixel requires more than 320 Mbytes/s of data transfer and 4 GFLOPS (billion operations per second). Such high speeds and massive amounts of data require a dedicated I/O link to the processing node (see Figure 1). Furthermore, a real-time application requires determinism on the I/O bus, which is not available on the standard buses. Dedicated parallel processing systems can overcome limitations of host bus speed and I/O throughput bottleneck and offer more performance per unit cost compared to a sequential supercomputer.

Parallel computing is the second fastest growing market next to network computing. In 1991, about 50% of the supercomputers sold were parallel computers. By 1996, approximately 70% of all supercomputers will be parallel computers [1]. A similar trend is emerging in minisupercomputers, superminicomputers, and workstations. Cray and IBM are targeting a performance of 1 TFLOPS (trillion floating-point operations per second).

A parallel computer consists of a collection of processing elements that cooperate to solve a problem by working simultaneously on different parts of the problem (similar to the work sharing done by the farmer's sons). The difference between the conventional Von Neuman computer, which consists of a single processor, and a parallel processing system is shown in Figure 2.

**Figure 2. Conventional Vs. Parallel Processing Systems**



(a) Von Neuman Computer With 'C40

(b) Parallel Processing System With 'C40

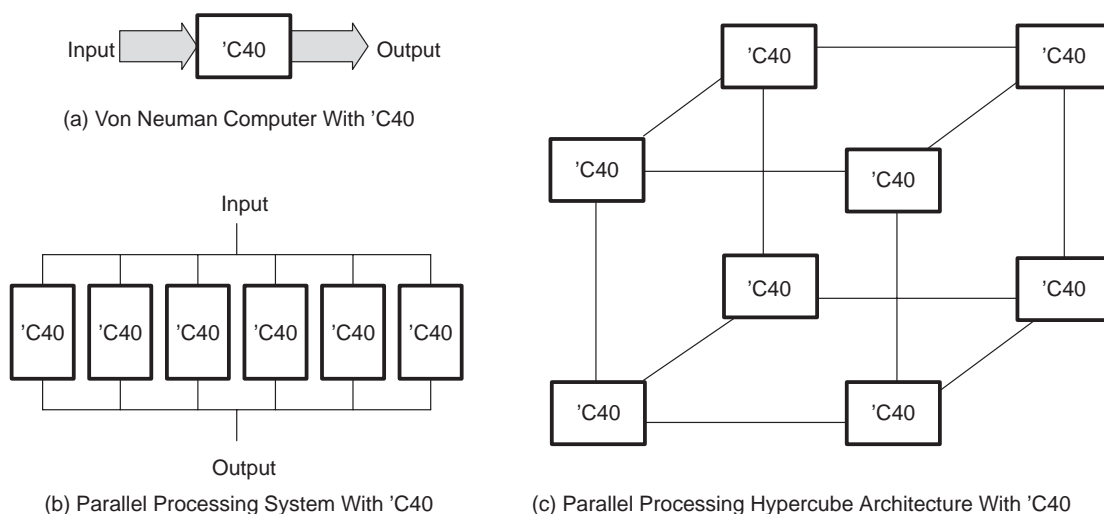(c) Parallel Processing Hypercube Architecture With 'C40

Figure 2(a) shows a conventional Von Neuman setup; Figure 2(b) shows a parallel processing system with multiple 'C40s; view (c) shows a hypercube architecture.

## Evaluating a Node for a Parallel Processing Architecture

A high-performance cost-effective parallel processing system for a nonspecific application must have nodes that meet these criteria:

- High processing speed,
- A large number of high-speed DMA-supported links,
- Easily configurable and incrementally expandable architecture,
- Ease in load balancing (even processing distribution over all the processors),
- Low cost per node or low cost per MIPS/MFLOPS/MOPS,
- Ease in programming via multitasking kernels and multiprocessing program support,
- High speed I/O, and
- Such development tools as compilers, assemblers, linkers, simulators, and emulation hardware.

### Why 'C40 as a Node

The selection of a node is critical for a parallel processing system. Many high-speed processors, such as the 80486, i860, and 68040, are capable of number crunching. None, however, offer parallel processing support. Some designers opt for a custom silicon solution for a node. The TMS320C40 from Texas Instruments, as shown in Figure 3, meets all the above defined criteria and is ideally suited for parallel processing.

**Figure 3.  TMS320C40 Parallel Processor Block Diagram**
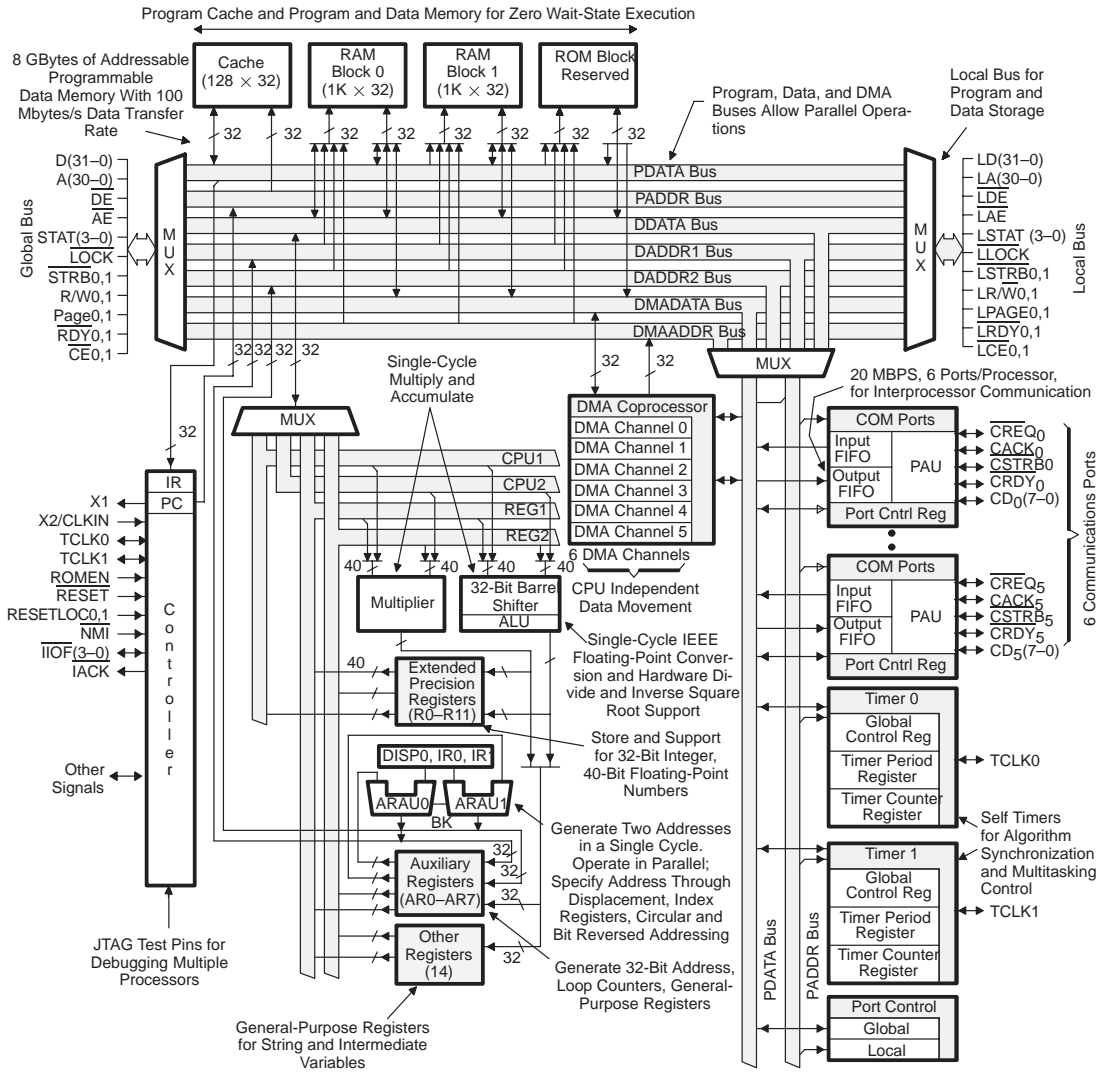


Texas Instruments, along with DSP board vendors, realized that over half of its customers were using multiple TMS320C30/31 DSP processors. The processors communicated via shared memory or from serial port to serial port. Such a configuration required additional hardware, software, and, most of all, degraded processor performance. The 'C40 was designed specifically to facilitate system configurability and provide massive parallelism with very little or no glue logic and without additional software overhead. The instruction set of the 'C40 is compatible with the popular 'C30 DSP processor. The parallel processing capability of the 'C40 [2], as illustrated in Figure 3, makes the 'C40 ideal for a parallel processing node and particularly for scientific and DSP applications. Table 1 summarizes the 'C40's key parallel-processing features.

**Table 1. 'C40 Parallel Processing Features**

| Criteria | Description |
| --- | --- |
| Node Speed | 40 to 50 MFLOPS, 275 MOPS (up to 11 operations per cycle throughput), 40/32-bit single-cycle floating-point/integer multiplier, hardware divide, and inverse square root support. |
| Node Link | Six 20-Mbyte/s node-to-node communication ports. Configuration for parallel processing topologies via the communication ports:<br><br>a. mesh up to three dimensions<br>b. pyramid of any dimension<br>c. hypercube of up to six dimensions |
| DMA | Six-channel DMA coprocessor supports 20-Mbyte/s transfers at each communication port. Compiler and operating system support for message passing and processor-to-processor communication. |
| Expandability | Up to six links with no glue logic.<br>Two identical external buses (32 bits data, 31 bits address).<br>Shared memory support. |
| Cost per Node | $500 to $1000 per node at 40 MFLOPS. |
| I/O per Node | 320 Mbytes (100 Mbytes/s on global bus, 100 Mbytes/s on the local bus, and 120 Mbytes on the six communication ports). |
| Programming Tools | ANSI C compiler, ADA, simulator, off-the-shelf boards for PC and VME, off-the-shelf DSP libraries. |
| Operating Systems | Currently supported by five manufacturers. See Conclusion section for details. |
| Hardware Debugging Tools | Debugging of multiple 'C40s on the Sun SPARCstation (Sonitech's JTAG Brahma SBus) or the TI XDS510 emulator. |
| Manufacturer Support | Support from TI and over 50 board and software vendors. |

The TMS320C40 is the processing node for Sonitech's ISA, VME, and SBus family of board-level products. The architecture of these boards is discussed below in *Parallel Processing Topologies*.

## Parallel Processing Topologies

This section presents several architectures that improve processing speeds. A matrix multiplication example is discussed to demonstrate the application of parallel processing. Several parallel processing architectures, such as hypercube, pyramid, ring, and mesh of trees topologies are explained in detail.

### Approaches to Performance/Throughput Enhancement

In the Von Neuman computer model, the steps of decoding, fetch, execution, and memory operations are performed in a sequential manner. Only one task is handled at a time; hence, the core CPU is utilized only during the execution phase. Newer approaches, such as pipeline, multitasking, and vectors have increased computer speed and CPU utilization.

In a pipeline, instruction decoding, data fetch, and memory operations are done in parallel over multiple queues while the CPU is executing instructions. If the execution of individual tasks does not depend on the results of other tasks, the pipeline will be full. If a process is data-dependent and lacks the most recent values, the pipeline will be blocked. The 'C40 has four levels of pipeline: fetch, decode, read, and execute. In addition, the DMA adds transfer capability and performs as many as three operations per cycle. The 'C40 has a separate multiplier, ALU, address register modifier, loop counter, and delayed branch (which can support eight operations per cycle).

6

## Parallel Processing Taxonomy

There is a wide diversity in parallel processing terminology. Two classifications [6], one by Flynn and the other by Skillicorn, can be implemented with 'C40s and are discussed briefly. Flynn categorizes computers into three major types:

- **SISD (Single-instruction stream/single-data stream)**

  This is a Von Neuman computer. A single 'C40 can serve as an SISD computer. (See Figure 2.)

- **SIMD (Single-instruction stream/multiple-data stream)**

  Traditional SIMD machines have a primitive node with no large local program memory. A controller feeds all the nodes with the same instruction and executes the program synchronously and in lock step. The interconnection network from processor to processor and from processor to memory can be an array, mesh, hypercube, or any network. Figure 4 shows a 'C40-based SIMD architecture. Since the 'C40 has 2K words of on-chip memory (and possibly more local memory on the local bus), the program blocks can be transferred from the host or the controller. The next program block can be transferred via the DMA without CPU intervention; this minimizes communication overhead on the interconnection network. Furthermore, the need for broadcasting from the host to the individual 'C40 is reduced.

**Figure 4. SIMD General Architecture With 'C40s**



Memory Unit (MU)
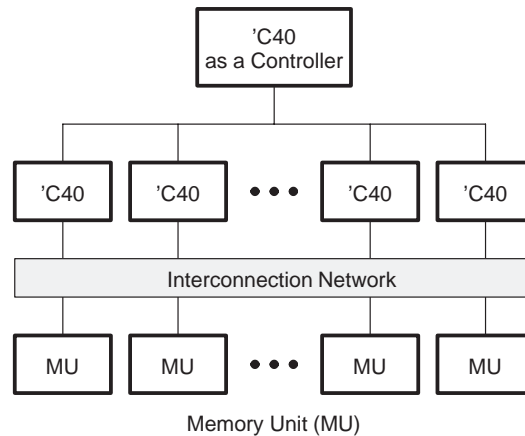
- **MIMD (Multiple-instruction stream/multiple-data stream)**

  In multiple-instruction stream/multiple-data stream architecture, as shown in Figure 5, each processor acts autonomously, executing different instructions. There are two categories of MIMD systems:

  – **Shared-memory or tightly coupled**

    The processors share the same memory, and there is a rigid protocol for reading, modifying, and writing the data in the shared memory.

  – **Distributed memory or loosely coupled machine**

    Each processor node has its own local memory (program and/or data); hence, there is no memory contention. Processors communicate via message passing. The interconnection network could be hypercube, ring, butterfly, mesh of trees, or another application-specific network.

The interconnection network is usually bus connected or directly connected. In bus connected systems, parallel memories, network interfaces, and device controllers are all connected to the same bus. In directly connected systems, the interconnection could be crossbar, partially connected graphs, or multistage networks.

**Figure 5.  Multiple-Instruction Stream/Multiple-Data Stream Systems**



MIMD With Shared Memory

MIMD With Local Memory
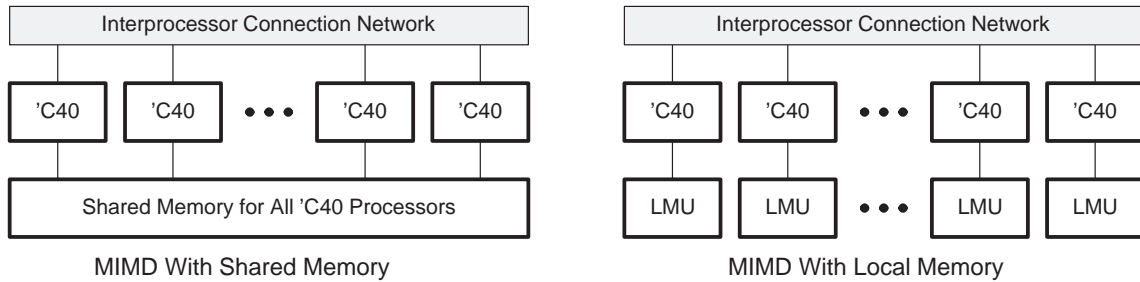
Flynn's taxonomy does not fully classify many new architectural configurations. Skillicorn's taxonomy of computer architecture is based on the functional structure of architecture and the data flow between its component parts, such as IP (instruction processor), DP (data processor), DM (data memory), IM (instruction memory), and SW (switching network). According to Skillicorn's classification, a total of 28 classes of computer architectural configurations are possible. Each classification is a function of the number of IPs, DPs, DMs, and IMs and their interconnection to each other. The interconnection also determines the method of message passing and/or message sharing and indicates the type of coupling of a particular parallel architecture. For example, Skillicorn's Computer Taxonomy class 14, known as a loosely coupled parallel processing system, has a total of n IPs and n DPs. Each IP is connected to each DP and to each IM; each DP is connected to each DM and to all other DPs. An n-dimensional hypercube, as shown in Figure 2(c), is an example of Skillicorn's class 14 parallel processing system.

## TMS320C40-Based AT/ISA and VME Architecture

Sonitech has developed VME and AT/ISA TMS320C40-based boards that are expandable from two nodes (100 MFLOPS) to hundreds of nodes, and memory that is expandable from 8 Mbytes to 768 Mbytes. This report discusses the architecture of these boards for readers who are designing their own target systems, integrating a system, or interfacing to the 'C40 global, local, or communication port bus. Figure 6 shows the SPIRIT-40 Dual with two 'C40s, the Quad-40 with four 'C40s, and the SPIRIT-DRAM with a large shared global memory. An ASM-M (application-specific module–memory) expansion bus brings the host bus from the Spirit-40 Dual to the Quad-40, SPIRIT-DRAM, or other custom boards.

**Figure 6.  AT/ISA and VME Architecture With the SPIRIT-40 Dual, SPIRIT-40 Quad, and SPIRIT-40 DRAM**
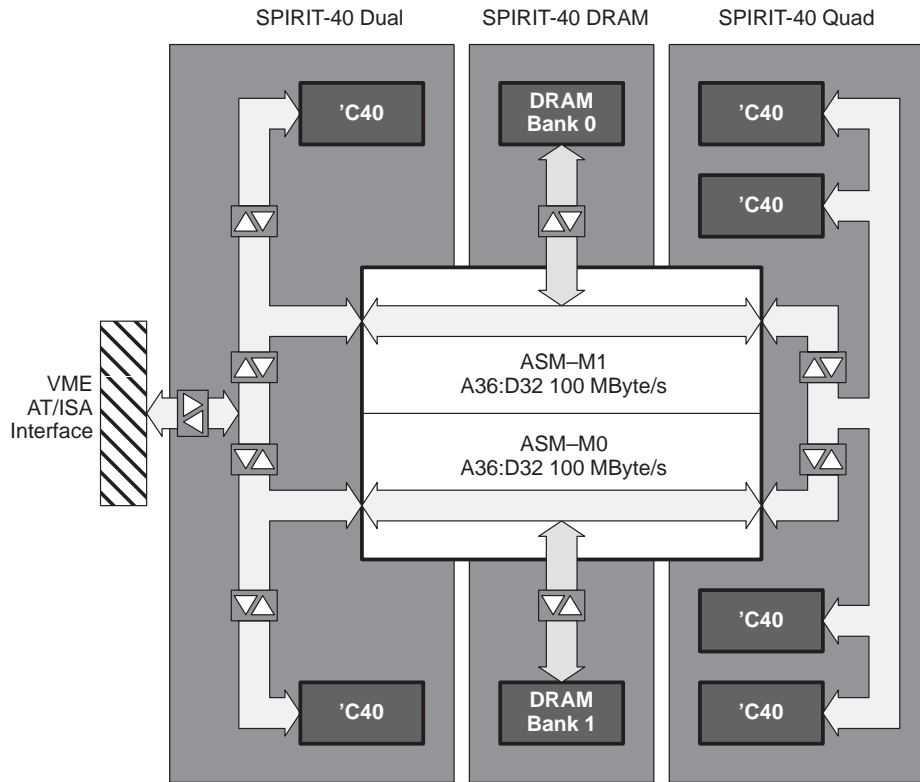


**Note**: For clarity, SRAM not shown.

This modular system (Figure 6) decouples the main processor card (SPIRIT-40) with a low-cost high-speed coprocessor unit (Quad-40) and with a large DRAM memory (SPIRIT-DRAM). Both the Quad and the DRAM boards feature no host bus (VME or ISA) interface for several reasons:

- The host bus is typically saturated (see Figure 1), and adding host interface will not increase the overall host to 'C40 system I/O,

- The ASM-M brings the host bus to the other two boards and to additional user-customized boards, and

- The configuration:
  - Saves 20 to 30% of board real estate,
  - Encourages the use of communication ports, as opposed to the host bus passing data and messages,
  - Reduces the cost of the Quad card (200 MFLOPS) to only $1000 more than the Dual (100 MFLOPS),
  - Uses minimal power per slot and is therefore more reliable, and
  - Requires no daughter cards.

9

A more detailed SPIRIT-40 VME architecture is shown in Figure 7, and major features of this architecture are highlighted in Figure 8. The two 'C40s are nearly identical, except that one has an RS232 interface and a boot EPROM. Attached to the 'C40 local bus is up to 4 Mbytes of local SRAM (can be as small as 256 bytes, or not used at all) and DSP control registers. The DSP control registers are specifically designed for configuring and programming massively parallel systems. Each register has a unique processor ID, the size and speed of local and global SRAMs, a processor boot mode (to boot from SRAM, PROM, or communication ports), the capability to reset and interrupt the direction and status of the six communication ports, and ASM-M port access and control. In a data flow architecture, where a large size data block is being routed in search of a processor with a large memory, the memory size register can be quickly read to determine whether that particular node has enough memory.
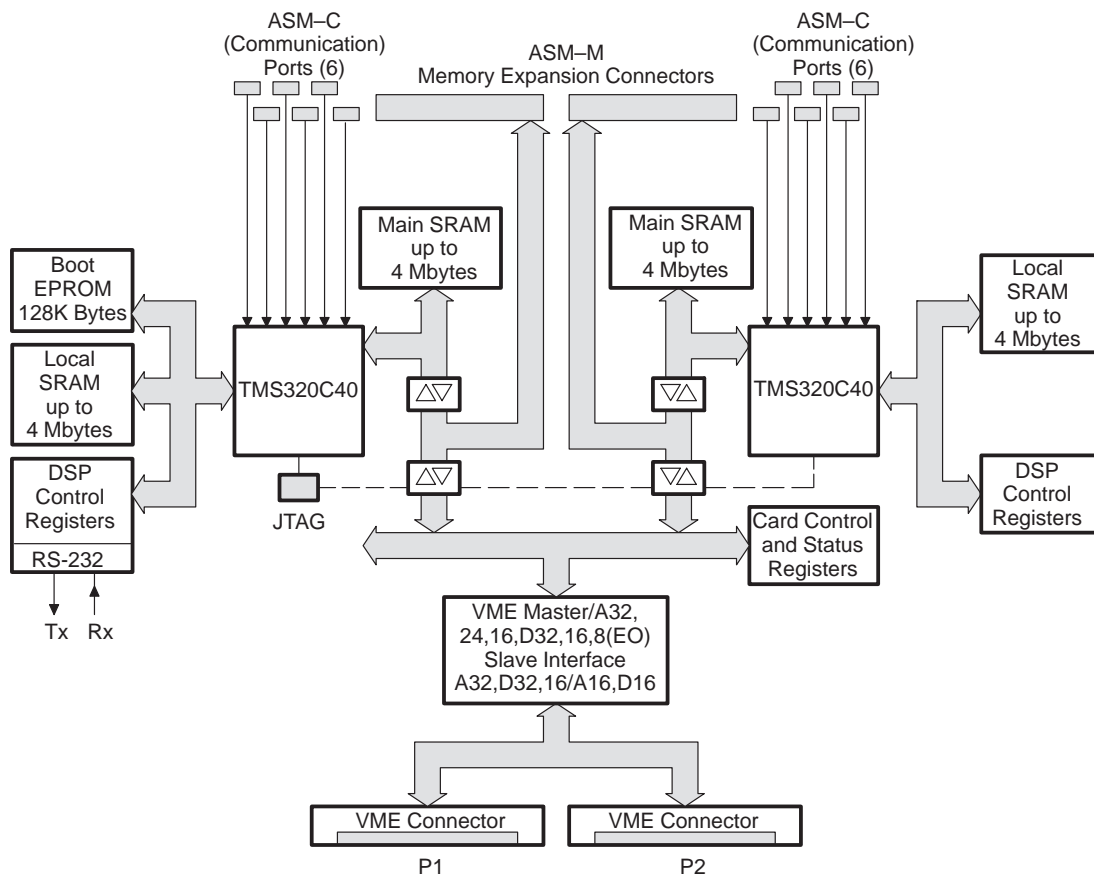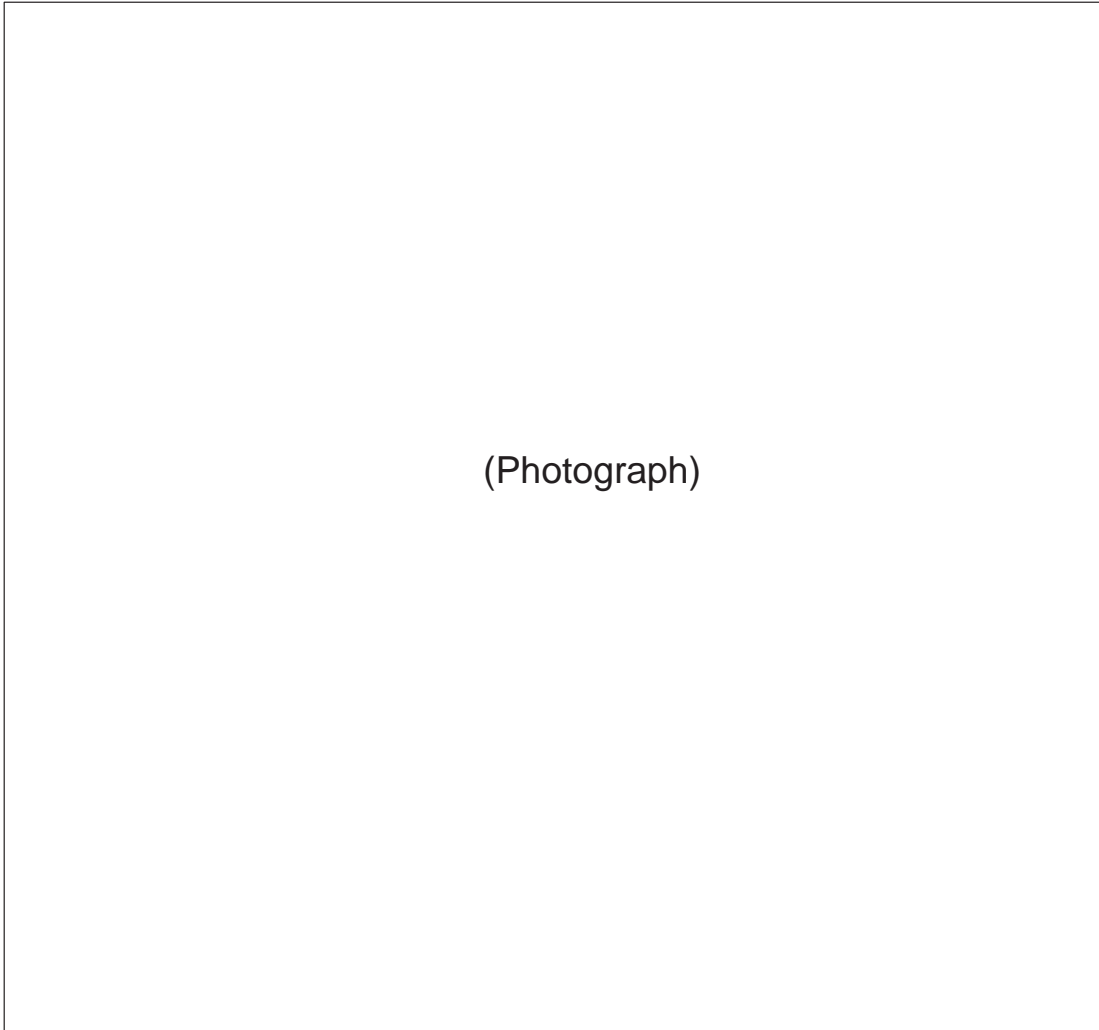
**Figure 7.  AT/ISA and VME Architectures**

**Figure 8.  Dual Processor SPIRIT-40 Major Features**



(Photograph)

1)  12 ASM-C Communication Port Connectors. Each 30-pin port connector is capable of 20-Mbyte/s transfers and has its own DMA controller. The 12 ports facilitate large processor arrays, data switches, and external I/O connections.

2)  ASM-M Communication Port Connector. This 240-pin connector provides two 50-Mbyte/s buses (32 bits wide, 32-bit address), one for each 'C40. The connectors make 256 Mbytes of DRAM available per 'C40 via a small interconnect board to an adjacent VME slot with Sonitech DRAM cards.

3)  Removable Faceplate. This faceplate unplugs to allow access to the ASM-M connector. In large systems (dozens of processors), the faceplate can be replaced by a user-designed panel to eliminate discrete cables.

4)  Memory SIMMs (single inline memory modules). These SRAM SIMMs provide zero-wait-state access for the 'C40. Up to 16 Mbytes can be installed on the board.

5)  TCLK. The TCLK lines connect to each 'C40's two timer-counter channels and can be used as inputs or outputs for controlling external equipment.

6)  JTAG. The industry-standard JTAG connector provides access to each 'C40's JTAG path for software development.

7)  VME Master/Slave Interface. This card may operate as a VME master, slave, standalone, or Slot 1 controller.

8)  Dual 40- or 50-MFLOPS 'C40 DSPs. The 'C40 supports a variety of development tools from Sonitech and other third parties, including a choice of five operating systems, C and FORTRAN compilers, and DSP libraries.

The main bus (or global bus) memory operates on a shared-access basis, giving the VME host and 'C40 access to the entire memory space on a cycle-by-cycle basis. This simplifies software development and speeds movement of large data arrays compared to systems that use a small dual-port RAM. Also on the 'C40 main bus (global bus) is the ASM-M expansion bus, which allows up to three cards to be connected. The VME bus has access to the ASM-M and can transfer data over this bus without interfering with the 'C40's access to its SRAM (notice the transceivers/buffers). One advantage of this implementation is that the host bus can transfer data to the ASM-M slave without impacting the 'C40 execution. The SPIRIT-40 VME transfers data at 20 to 30 Mbytes/s over the host VME bus and is a slot 1 bus master; that is, it reads and writes to the memory of other cards on the VME bus.

In addition to the local and global memory interfaces to the 'C40, six communication ports are connected to the front panel. To simplify interconnection to external communication port devices (such as data acquisition, video bus, or other custom interfaces), each communication port connector includes an external interrupt signal to the 'C40 and a reset signal that can be configured as an input or output. (See Figure 10.) A secondary connector is provided for access to the timer clock.

The RS232 can interface to any host machines (MAC, Sun, PC, etc.) for code downloading, debugging, and diagnostics. For stand-alone operations, the boot EPROM can boot not only the main 'C40, but also all the other 'C40s.

The coprocessor SPIRIT-Quad is shown in Figure 9. It interfaces with the ASM-M via the shared SRAM, and the global buses of each 'C40 can access this SRAM. The 'C40 processors on the Quad have the same type and size of memory and the same DSP control register as the processor on the Dual card. Such symmetry is highly desirable, as discussed later in this report, for many architectural configurations and algorithms. A total of 12 communication ports are brought out. There are three from each 'C40; the other communication ports are connected among themselves.

## Figure 9.  SPIRIT-40 Quad Processor Architecture
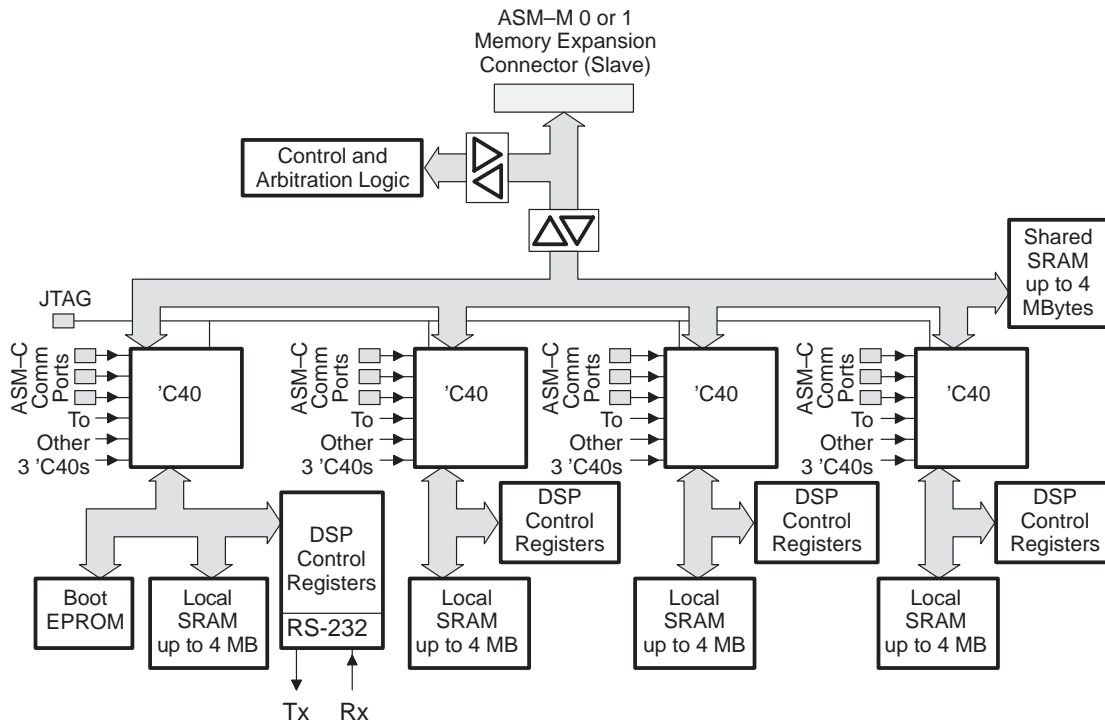


The SPIRIT-DRAM supports applications requiring massive amounts of global memory. Like the SPIRIT-Quad, the SPIRIT-DRAM has no host interface (only power and grounds are connected); the host bus is available via the ASM-M memory expansion connector. A total of 256 Mbytes of DRAM is available on a shared-access basis (one to three wait states) to all the 'C40s on the Dual board.

## System-Level Design Issues With the TMS320C40 Communication Ports

The TMS320C40 communication ports transfer data at 20 Mbytes/s. This communication capability makes the 'C40 ideally suited for large interconnect parallel processing networks. You should analyze several design issues, however, before selecting an interconnect topology and peripheral interface design. The 'C40 communication ports consist of 12 lines: eight data, one strobe, one ready for byte-transfer handshaking, one for token transfer, and one for token acknowledge. A communication port can be in either of two I/O states: output state if it has the token, or input state if it does not have the token. During power-up, three of the six communication ports have a token (output ports), and three don't have a token (input port). For bidirectional communication, the connected input and output communication ports pass the tokens back and forth.

### 'C40 Node Reset

Figure 10 shows a configuration with two 'C40 nodes, A and B, each with one communication link. The power-up default is that A-Link has the token and is an output; B-Link does not have the token and is an input. During bidirectional data transfers, A and B exchange tokens. If B-Link has the token, node A is reset; after the reset, both A-Link and B-Link will have the token. Both links then drive the data bus (if for more than 100 ns, the communication ports could be damaged).

13

**Figure 10.  Token State After Reset**
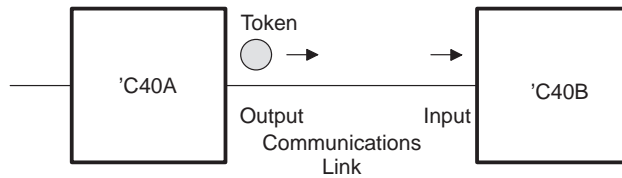


The SPIRIT-40 communication ports have signals that prevent communication link contention and simplify interface to peripheral devices.

In a large topology where multiple 'C40 links are connected to many other 'C40s, a reset to one 'C40 must immediately follow a reset to all 'C40s. Such a scheme has significant software and system-level ramifications. For example, all of the intermediate data and programs, including program and data download, will be lost and must be reinitialized. The problem is further complicated if a peripheral device is attached to a link and has an option to assert a reset to the 'C40.

The ASM-C (application-specific module–communication port interface), a superset version of the 'C40 communication port interface, guarantees no contention between the communication links. The bidirectional CTOKRES* (token restore) signal resets all the 'C40 links, is received by all the 'C40's in the cluster, and is connected to each 'C40's NMI (nonmaskable interrupt). Once the 'C40 enters the NMI routine, it can select reset or no reset. The CAUXRES* and CINT* I/O signals are provided for non-'C40 reset. These signals will not reset the 'C40 or a communication port token interface. They can reset a remote communication-link-connected peripheral device or one 'C40 interrupting another 'C40. When a remote device or another 'C40 asserts this signal, the 'C40 that receives the signal is interrupted and can take appropriate action in software.

## Design Suggestions for a Large Topology

- **Use redundant links to avoid bidirectional communication port I/O**

    The 'C40 has six communication ports. If a four-processor configuration with single links, as in Figure 11, is required, the remaining links can be used so that only unidirectional transfers are necessary on each link. Such a configuration permits reset of a single 'C40 without cluster reset. For example, a 3-D hypercube (three links per node) can be configured for six unidirectional links (dashed lines).

**Figure 11.  Four-Node Configuration With a Single-Link Interface and Unidirectional Transfers**
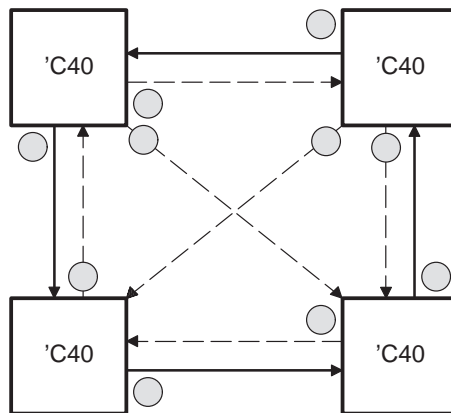
- **Partial cluster reset**

  For applications that cannot tolerate a full system reset, you can configure the hardware and software for partial cluster reset. For example, in the two-dimensional mesh shown in Figure 12, the thick lines represent a second link. This way, each link is maintained unidirectionally in the horizontal direction and so is not affected by a reset. A reset at any one node needs to propagate in the vertical direction only, not in the horizontal direction.

- **Use Buffered Communication Ports**

  Buffering the communication ports will reduce the transfer speeds and, depending on the design, usually allow only unidirectional transfers. The advantage is that the communication port links can be significantly longer than 6–12 inches.

**Figure 12.  A Two-Dimensional Mesh Configuration**



```
——  = Bidirectional link
 x  = 'C40 under reset
—— — = Column to be reset
——► = Unidirectional link
```
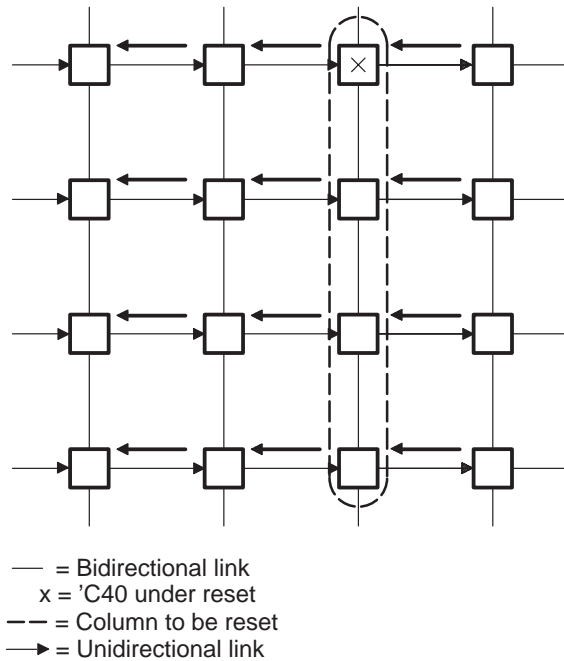
## Matrix Multiplication Application of Parallel Processing

Parallel processing has a wide range of applications, including the following:

- embedded real-time
- image processing
- feature extraction
- motion detection
- speed estimation
- thermographic analysis
- speech recognition
- spatial and temporal pattern recognition
- vision

- weather
- seismology
- defense
- radar
- sonar
- scientific
- medical
- industrial
- communications

One commonly used algorithm is matrix multiplication. In general, if A is a matrix of $m \times n$ and B is a matrix of $n \times k$ dimensions, respectively, then a mesh of size $m \times k$ is required to compute matrix-matrix multiplication. In our example, $m = n = k = 2$.

The example below and Figure 13 show an example of matrix multiplication. Let the nodes of the mesh be labeled like a matrix element (row,column), that is, nodes (1,1), (1,2), (2,1), and (2,2)

$$A \times B = \begin{vmatrix} 2 & 1 \\ 4 & 3 \end{vmatrix} \times \begin{vmatrix} 7 & 8 \\ 5 & 6 \end{vmatrix} = C = \begin{vmatrix} 19 & 22 \\ 43 & 50 \end{vmatrix}$$

Step 1.  'C40(1,1)
  d=A(1,1)*B(1,1)=2*7=14

Step 2.  'C40(1,1)
  C(1,1)=A(1,2)*B(2,1)+d=1*5+14=19
  'C40(1,2)
  e=A(1,1)*B(1,2)=2*8=16
  'C40(2,1)
  f=A(2,1)*B(1,1)=4*7=28

Step 3.  'C40(1,2)
  C(1,2)=A(1,2)*B(2,2)+e=1*6+16=22
  'C40(2,1)
  C(2,1)=A(2,2)*B(2,1)+f=(3*5)+28=43
  'C40(2,2)
  g=A(2,1)*B(1,2)=4*8=32

Step 4.  'C40(2,2)
  C(2,2)=A(2,2)*B(2,2)+g=(3*6)+32=50

At the end of the algorithm, each 'C40 node designated as 'C40(i,j) contains element C(i,j). Reference [3]explains parallel implementation of matrix multiplication using both shared and distributed memory approaches. The features for the 'C40 architecture bring high efficiency, low-communication overhead, and almost perfect speed-up to this implementation.

**Figure 13. Matrix Multiplication**
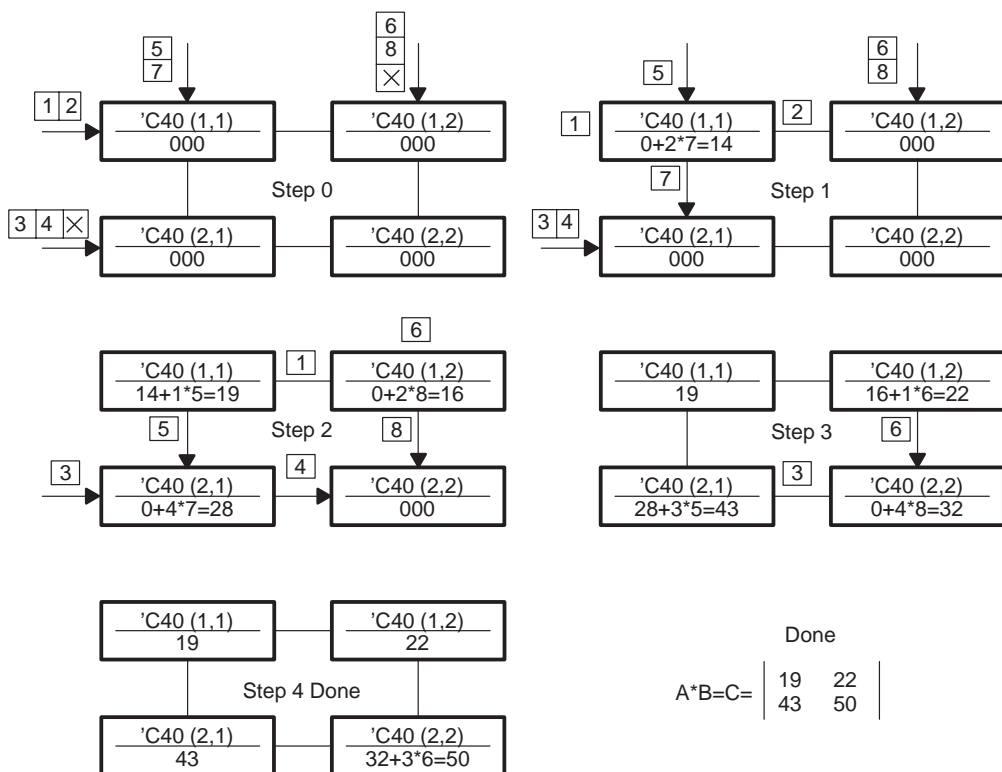


## Parallel Processing Architecture Topologies

This section discusses the hypercube, pyramid, ring, mesh, and mesh of trees architectures and their properties.

### Hypercube and Its Properties

The hypercube is a configuration of loosely coupled parallel processors based on the binary n-cube network. A significant difference between hypercube and most other parallel architectures is that its modes use message passing instead of shared variables or shared memory to communicate. These are the key advantages of a hypercube architecture:

- **Powerful Network**: Easily mapped and efficiently implemented.
- **Small Diameter**: Shortest path separating the farthest two nodes.
- **Alternate Paths**: Alternate paths for all nodes to minimize congestion.
- **Scalability**: Flexibility in expanding and contracting the network without significant hardware and software modification and cost.
- **Fault Tolerance**: Robust when nodes and links malfunction.

The 'C40's six communication ports can be connected at 20 Mbytes/s to form a six-dimensional hypercube (64 nodes). In addition, the 'C40's large amount of addressable local, global, internal, and cache memory, coupled with its CPU performance, DMA, and pipeline, permits a large subprogram to be implemented efficiently on one node. This reduces the node-to-node communication overhead.

17

An n-dimensional hypercube contains $2^n$ nodes. Nodes are connected directly with each other if their binary addresses differ by a single bit. In an n-dimensional hypercube $H_n$, each processor is directly connected with n neighboring processors with n-bit binary addresses in the interval 0 to $2^n-1$. The processing elements are placed at each vertex of the hypercube. Adjacent nodes are connected directly with each other if and only if their binary addresses differ by a single bit. One commonly noted disadvantage of a p-cube computer (p=number of nodes) is its requirement for large numbers ($\log_2 p$) of input/output ports per node.

If a hypercube has fewer than six dimensions, the extra links from each 'C40 can be used as secondary links for high reliability and full duplex communication and to decrease the node-to-node distance. In special cases where two or more paths of equal distance are to be connected, priority is given to those paths whose planes have complementary MSBs; this yields a system with a minimum diameter. Figure 14 shows two- and three-dimensional hypercubes without extra links.

### Figure 14.  Two- and Three-Dimensional Hypercubes



(a) Two-Dimensional Hypercube

(b) Three-Dimensional Hypercube
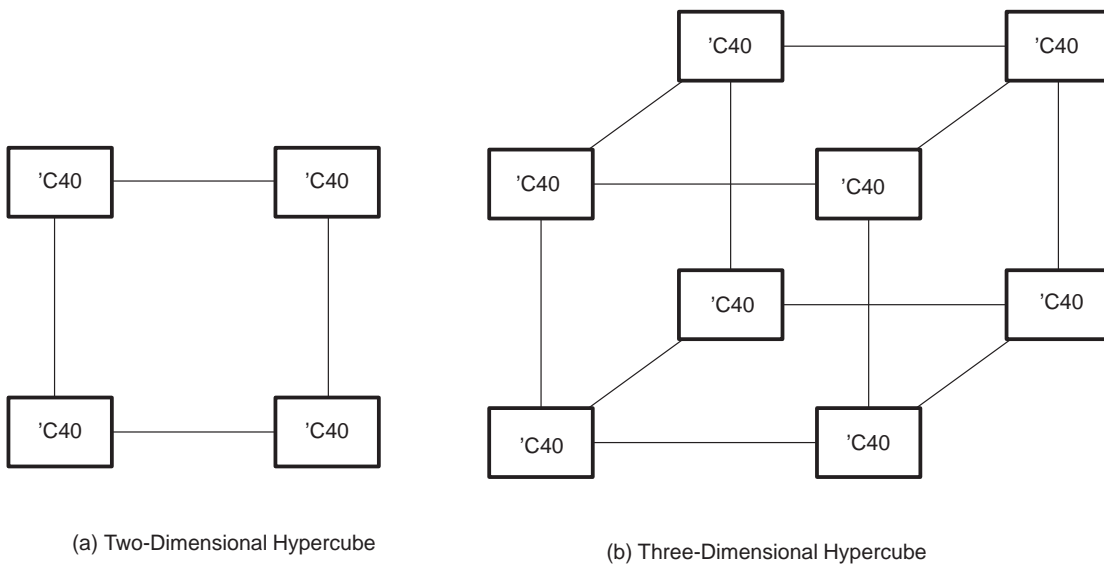
You can also create a hypercube of up to ten dimensions by combining two 'C40s as one node (see Figure 15). One of the two 'C40s is the number-crunching processor; the other 'C40 is used for communication, control, task scheduling, data decomposition, and, if required, number-crunching support for the first processor. This configuration has a peak performance of 100 GFLOPS. The SPIRIT-40 Dual processor is ideal for this dual-processor/single-node configuration.

**Figure 15. Ten-Dimensional Hypercube Configured With Dual Processor—Single Node**
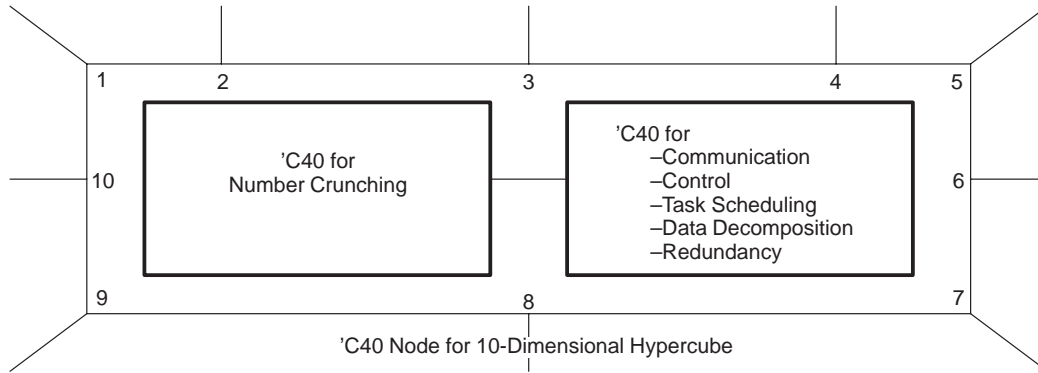


## Hypercube Node Assignment

Hypercube nodes are assigned an address with respect to the Reflexive Gray Code (RGC). RGC is important because it uniquely defines adjacent nodes and node-to-node distances. RGC can be expressed mathematically as:

$$G_{n+1} = \{(0)G_n, \ (1)(G_n^r)\} . \tag{1}$$

where $G_n$ is the RGC of order n, $G_n^r$ is the reflection of $G_n$, $0(G_n)$ is the insertion of 0 as a MSB, and $1(G_n)$ is the insertion of 1 as a MSB.

The following are RGCs for n = 2, 3, and 4.

- One-dimensional

$$G_1 = 0; \text{and } G_1^r = 1 \tag{2}$$

- Two-dimensional

$$G_1 = 0, \ 1, \ \text{and} \ G_1^r = 1, 0; \text{therefore}, G_2 = \mathbf{0}0, \mathbf{0}1, \mathbf{1}1, \mathbf{1}0 \tag{3}$$

Similarly,

- Three-dimensional

$$G_3 = \mathbf{0}00, \mathbf{0}01, \mathbf{0}11, \mathbf{0}10, \mathbf{1}11, \mathbf{1}10, \mathbf{1}00, \mathbf{1}01 \tag{4}$$

- Four-dimensional

$$G_4 = \mathbf{0}000, \mathbf{0}001, \mathbf{0}011, \mathbf{0}010, \mathbf{0}111, \mathbf{0}110, \mathbf{0}100, \mathbf{0}101,$$
$$\mathbf{1}111, \mathbf{1}110, \mathbf{1}100, \mathbf{1}101, \mathbf{1}000, \mathbf{1}001, \mathbf{1}011, \mathbf{1}010 \tag{5}$$

In addition to the node addresses as described above, the link connections must be assigned. The number of links or edges in a hypercube are $n \times 2^{n-1}$. On the 'C40-based n=6 hypercube, 192 cables will be required. Port connections are assigned via an iterative algorithm.

19

### Six Properties of the Hypercube

Six properties [8, 11, 14] are unique to the hypercube and can be applied to task allocation, data decomposition, communication, and architectural expansion and contraction:

1. "An n-dimensional hypercube can be constructed recursively from a lower dimensional cube."

   This property makes a hypercube a modular parallel architecture; it allows you to build a higher dimensional hypercube from lower dimensional hypercubes. With 'C40s as nodes, you can always upgrade the hypercube a minimum of six dimensions or higher by combining two 'C40s into one, as shown in Figure 15.

2. "Any n-cube can be tiered in n possible ways into two (n–1) subcubes."

   Due to this property, hypercube can divide larger data size into smaller data size, assign the tasks to different subcubes in any order, and combine the results.

3. "There are $n!2^n$ ways of numbering the $2^n$ nodes of the n-cube."

   It is not necessary to assign node addresses in any specific order. Since the node addresses are RGC, you can assign the starting addresses to any node. This is especially useful when the data input node changes from one node to another; to implement the same algorithm, just reassign the node address rather than change the program or the physical link connection.

4. "Any two adjacent nodes A and B of an n-cube are such that the nodes adjacent to A and those adjacent to B are connected in a one-to-one fashion."

   In the three-dimensional cube shown in Figure 16, two adjacent nodes, A and B, are chosen arbitrarily. A has a1, a2, a3 as its adjacent nodes, and B has b1, b2, b3 as its adjacent nodes. Note that the pairs (a1, b1), (a2, b2), and (a3, b3) are also adjacent. If this process is repeated until all the nodes are marked either a or b, then exactly half the nodes are marked a, and the remaining half are marked b. All the nodes are linked in a one-to-one fashion (and are adjacent) like nodes A and B.

**Figure 16. Three-Dimensional Cube**
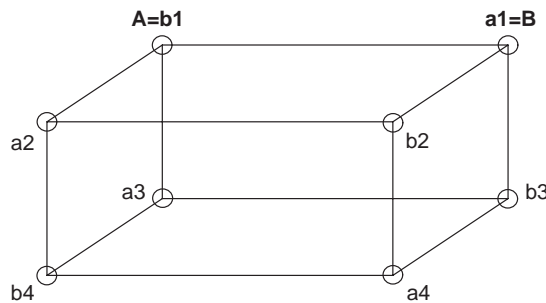


5. "There are no cycles of odd length in an n-cube."

   This property simplifies message passing, which is the heart of the loosely coupled MIMD hypercubes. For example, a message traveling from node A back to node A will pass through an even number of links. Parity is positive (+tve) if the number of 1s in a binary address of a node is even; if the number of 1s is odd, the parity is negative (–tve).

6. "The n-cube is a connected graph of diameter n."

   When a message travels from one node to any other node, the shortest distance will not exceed the diameter, which is equal to the hypercube dimension, n. Additionally, there are n different (nonoverlapping) parallel paths (see node A and node B paths in Figure 17) between any two nodes. These parallel paths are equal if A and B are the farthest nodes. The message must pass through n links or less in an n-dimension hypercube. In other words, in an n-dimension hypercube, a message needs to pass through only n–1 (or less) nodes to get to the intended node.

**Figure 17. Hypercube Architecture With Parallel Paths**
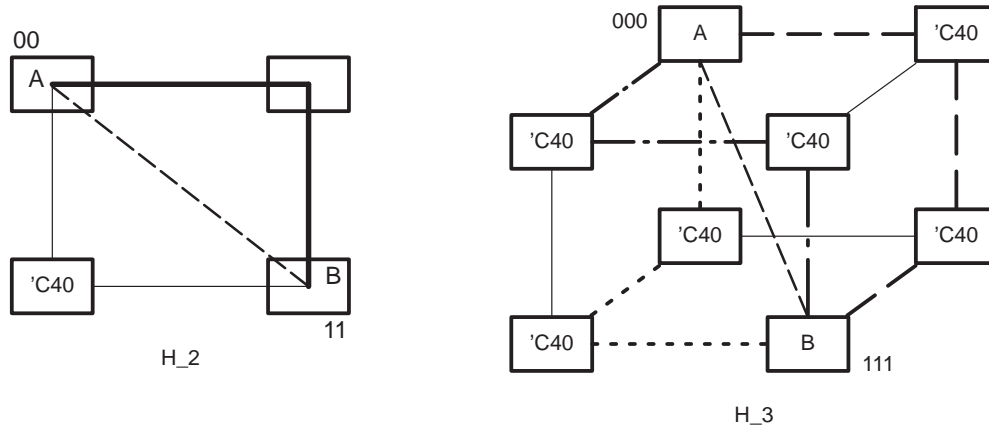


### *Distinguishing Hypercube From Similar Configurations*

Many architecture configurations such as wraparound mesh or wraparound tori can be mistaken for hypercubes. The following theorem differentiates hypercubes from other similar architecture configurations.

Theorem: A graph G with a total number of V nodes and E edges is a hypercube if and only if:

- V has $2^n$ nodes.
- Every node has degree n.
- G is connected in a diameter of n (see hypercube property six, above).
- Any two adjacent nodes A and B are such that the nodes adjacent to A and those adjacent to B are linked in a one-to-one fashion (see hypercube property four, above).

This theorem is proved by induction and discussed elsewhere, but it can be verified easily by applying the theorem to a mesh with eight nodes and a hypercube of three dimensions. In summary, the hypercube is one of the most versatile and efficient networks. It is well suited for both special-purpose and general-purpose tasks and can easily be implemented with the TMS320C40 DSP.

## Mesh Topologies

### *Simple Mesh*

A two-dimensional mesh is the simplest parallel processing topology. Its nodes are located at the intersection of rows and column links. Stacking two-dimensional meshes and linking all the nodes in the vertical plane yields a three-dimensional mesh (see Figure 18). The 'C40 is perfectly suited for a 3-D mesh in which four communication ports are linked to neighbors in the horizontal plane, and the remaining two are linked in the vertical plane. The edge nodes require only four links, and the corner nodes require three links. Extra edge and corner links can be used for data input/output, redundancy, and expandability. The mesh diameter is of the order log n, where n is the total number of mesh nodes.

**Figure 18.  Three-Dimensional Mesh**
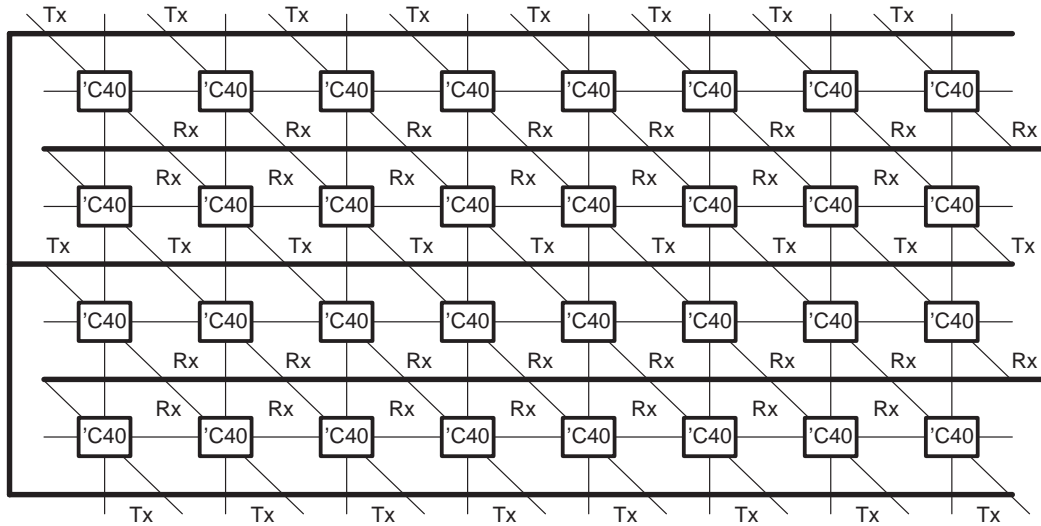


### Mesh of Trees

Mesh of trees architecture gives a higher overall performance than an array ORing topology. The n × n mesh of trees [14] is constructed from an n × n mesh by adding processors and links to form a complete binary tree in each row and each column. In Figure 19, the mesh nodes are square and numbered from 00 to 15. A 'C40 row tree node is added to the mesh node 00 and node 04. Similarly, mesh nodes 08 and 12 are linked to another 'C40 tree node, and both of these tree nodes are linked to a 'C40 apex node to complete the binary row tree. After a complete binary row tree is completed for all the mesh nodes, columns are formed in the same way.

**Figure 19. Mesh of Trees Architecture**
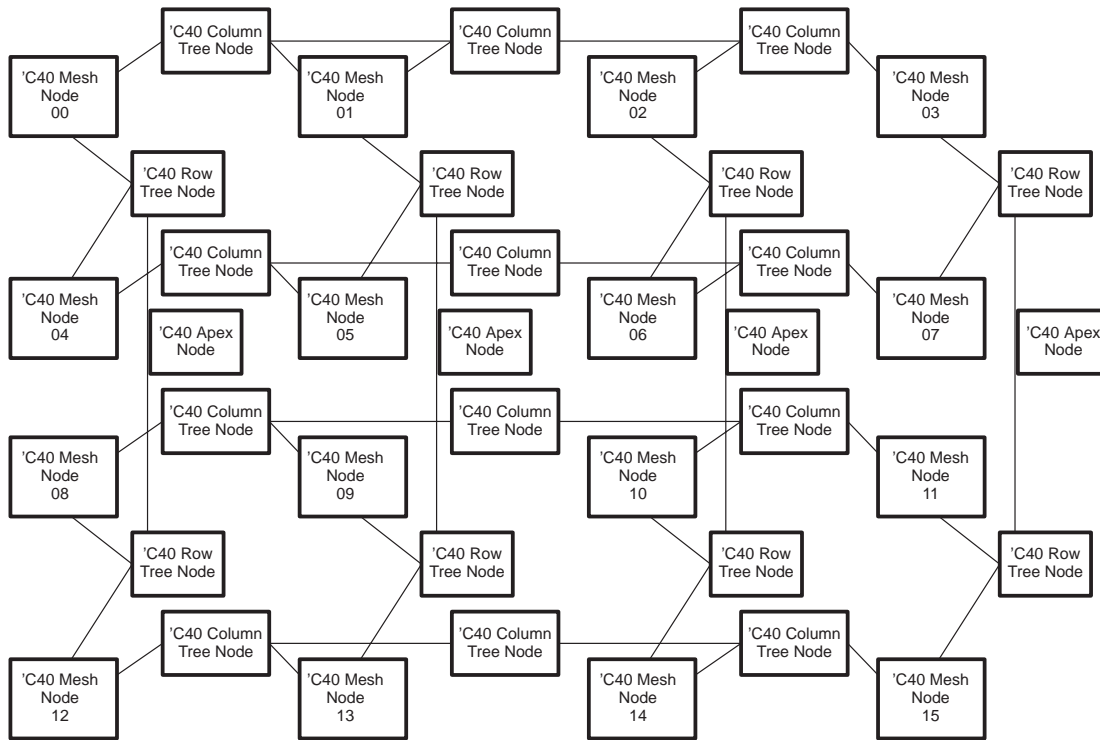


The tree leaves are the original $n^2$ nodes of the mesh. Overall, the network has $3n^2 - 2n$ processors. The leaf and root processors have two links, and all other processors have three links. The diameter of $n \times n$ mesh of trees is $4 \log n$. The mesh of trees also exhibits the recursive decomposition property, which is useful for parallel computations, for designing efficient layouts, and for fabricating a larger dimension mesh of trees from smaller dimension meshes of trees. This property is also helpful in implementing identical computations at low levels using the subnetworks, and for high-level global communication using the root processors. As the size of the network increases, the number of ports for communication increases linearly; this disadvantage is similar to that of hypercubes.

### *Pyramid Architecture and Its Properties*

A pyramid is a repetitive structure whose building block consists of the basic unit shown in Figure 20. Pyramids are widely used in image processing, where large data is segmented and decomposed.

**Figure 20. Pyramid Building Block**
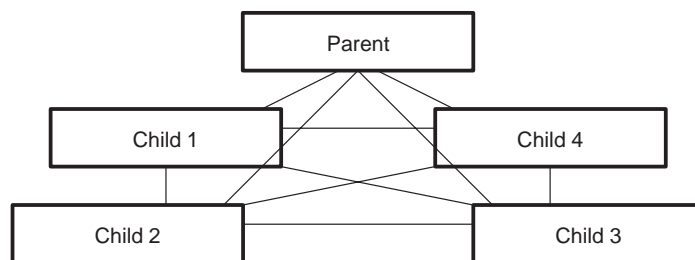
There are five 'C40 nodes in the building block of the pyramid. The 'C40 node on the top is called the parent of the four lower-level child nodes. Each parent can have only four children, and all the children are connected in a mesh. Each level of a pyramid is a mesh, and the bottom level or the level with maximum nodes ($2^n \times 2^n$) is called the *base* level. The ratio of nodes in a lower level to the node(s) in the adjacent upper level is 4:1. The topmost level ( level 0) is called the *apex* of the pyramid, and it has only one node ($2^0 \times 2^0=1$).

A two-level 'C40-based pyramid (see Figure 21) has 21 nodes (one apex node requires four links, four nodes at level 1 require seven links, and 16 base-level nodes require a maximum of five links). Since the 'C40 has a maximum of six communication ports, some nodes at level 1 will not be symmetrical. To compensate, a "dual processor single node," as shown in Figure 15, is suggested for these nodes. The other option is to include a one-node-to-two-node multiplexer.

## Figure 21. Two-Level Pyramid
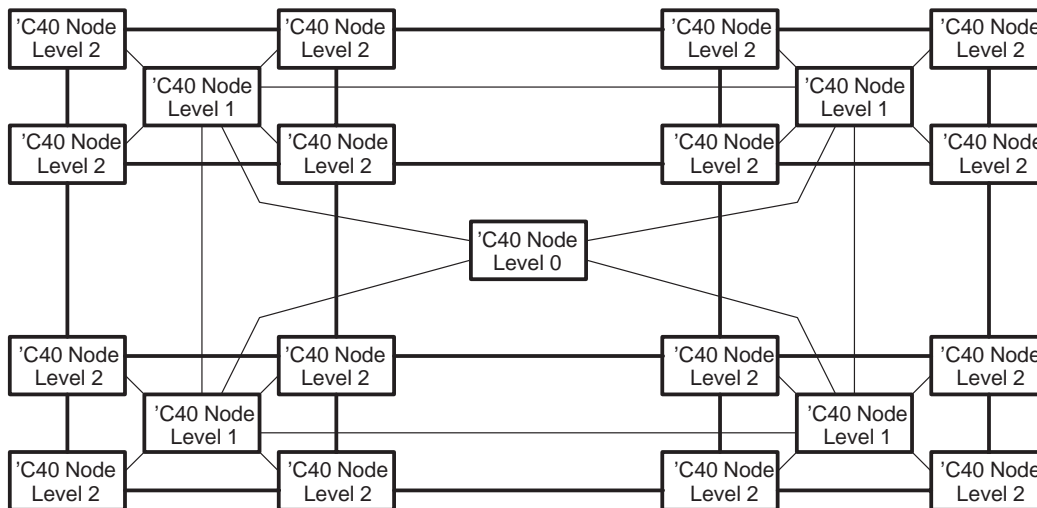


The 'C40 offers several advantages when you build a pyramid. The concurrent DMA and CPU operations over the links reduce communication overhead for large amounts of data. The apex node has two free links that can be used for data I/O. Also, the children can pass data among themselves without slowing down the parent nodes. This is an advantage in image processing, where data decomposition and task scheduling are common.

The Quad-40 board is a complete set of four children. To build a two-level pyramid, you need boards for the base level (a total of 16 'C40s), another board for level 1, and a SPIRIT-40 dual board to complete the pyramid structure.

### Ring Architecture

A ring architecture and its variations, as shown in Figure 22, consist of a number of processors connected in an array fashion. When the last processor of an array is connected back to the first processor (wraparound array), a ring is formed. Ring architecture offers several advantages over a simple array in terms of computation and communication times. The communication protocols are well established and simple. When a mode or link malfunctions in ring architecture, however, messages must be rerouted in the reverse direction from the break point. Such rerouting taxes each processor's CPU and increases the overall bandwidth on the remaining links.

The 'C40 processor node can increase the ring architecture reliability and link bandwidth by 300%, as shown in Figure 22a. The most versatile extension of ring architecture is the vertical stack of rings organized to form a cylindrical architecture. The cylindrical architecture offers two links for increased reliability in the horizontal plane and r number of vertical parallel paths, as shown in Figure 22c, where r is the number of nodes in any ring layer. The cylinder also provides for inputs at the top and outputs at the bottom. Cylindrical structures are highly modular because it is possible to increase the diameter of the ring and the height of the cylinder.

**Figure 22.  Ring Architecture**



a. 300% Redundant Ring

b. 200% Redundant Ring With I/O Ports

c. Modular Cylindrical Architecture
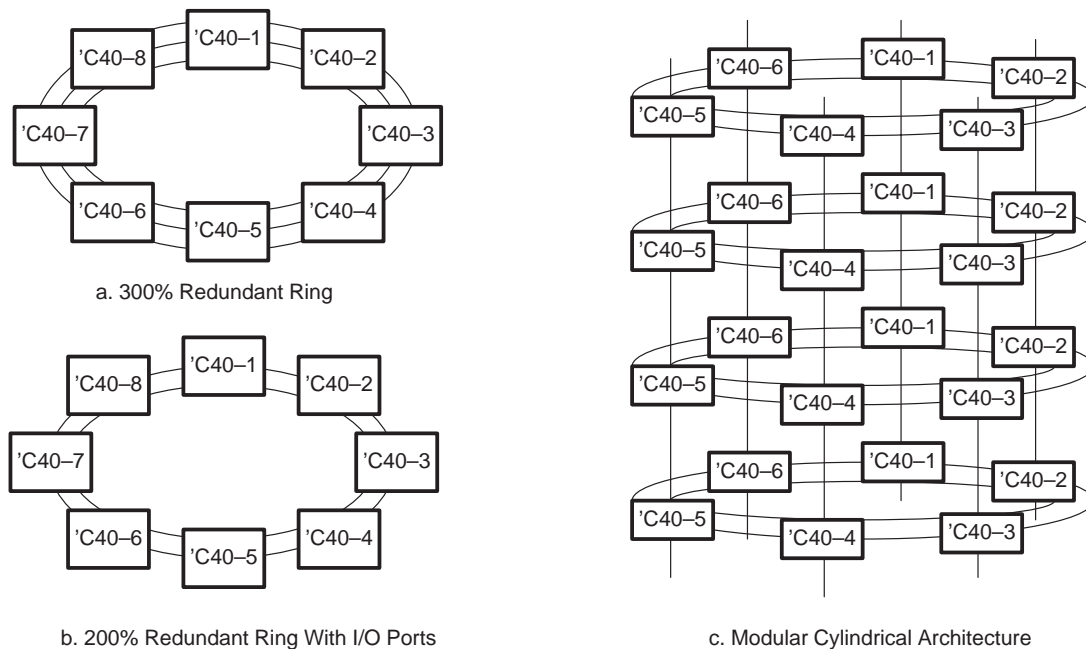
## Reconfigurable Massively Parallel Processors

Massively parallel computers require many links, and the link-to-processor ratio differs, depending on the selected architecture. For example, the 'C40 links support hypercubes with a maximum of six dimensions and pyramids with only one level. To overcome the link limitation, massively parallel systems require reconfigurable architecture. Reconfigurable computers, as the name implies, can be reconfigured from one architecture to another without any physical modification in processor-to-processor connections. The computers are reconfigured via a link switch, which multiplexes one link to many or many links to many others. This class of machines offers high fault tolerance and overall lower cost systems if the cost of a link switch is less than adding additional links to the processor.

Switch configuration can be performed before the processing begins (nonreal-time), deterministically, or dynamically as the switch passes the packets to its destination. Hardware switch reconfiguration requires physical switches, and the following factors must be evaluated:

- Total number of paths that enter and exit from the switch (degree)
- Number of possible configurations for the switch (configuration settings)
- Number of lines entering the switch from each direction (data path width)
- Number of different paths that the switch can simultaneously connect (crossover capability)

Figure 23 shows a $2 \times 2$ switch in which any one link can be connected to any one of the other three links. This switch reconfigurability offers three processor-to-processor connections with only one link per processor. For example, the 'C40's six ports can be physically increased to 18 links (18 adjacent nodes or an 18-dimension hypercube) via the $2 \times 2$ switch on each port.

**Figure 23.  A Two-by-Two Switch With Four Possible Switch Settings**
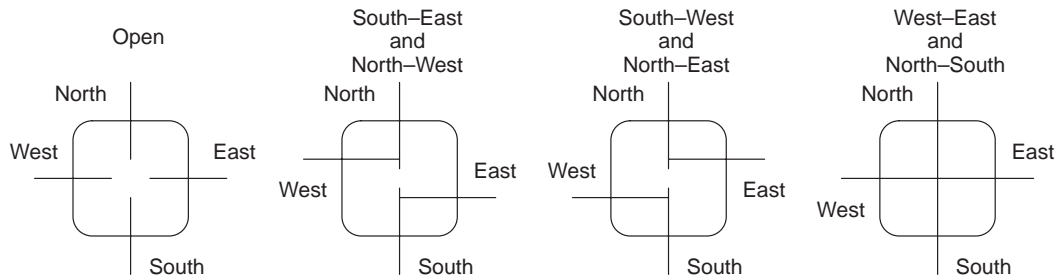


The switch connection mechanism can be controlled by circuit switching or packet switching. In circuit switching, the links are pre-established; in packet switching, the packet header carries the destination address. Switches are cost-effective in implementing massively parallel processors.

## Software Approach to Massively Parallel Systems

The software approach for reconfiguration of interconnection networks is based on the graph embedding principle. Graph embedding, also known as mapping, and other techniques based on graph embedding, such as reshaping, direct embedding, embedding by graph decomposition, and many-to-one embedding can be found in related literature.

## Links for Massively Parallel Processing

Massively parallel systems require a large number of links. Large numbers of links can cause problems of electromotive force (emf) generation, signal degradation due to large link distances, cross-talk, ground loops, security, and less bandwidth speed. If a bus architecture is used instead of links, then the shared bus also lowers the system speed and requires extensive control and arbitration logic.

In the future, fiber-optic-based 'C40 communication port links will overcome the problems. The new fiber-optic link uses low power; has a single IC with transmitter, receiver, serial-to-parallel conversion, clock generation, and recovery functions; and is fully compatible with the emerging FDDI and other upcoming standards.

# Benchmarks, Analysis, and Data Decomposition

## Benchmarks

This section summarizes several benchmarks for evaluating multiprocessing architecture performance:

- **MFLOPS**

  This theoretical number represents the maximum number of arithmetic operations or instructions that can be executed by a device and is expressed as millions of floating-point operations per second (MFLOPS). The 'C40 can perform 40–50 MFLOPS. In some implementations such as filtering, it is possible to achieve 99% of theoretical MFLOPS. The factors that decrease throughput are memory contention, input/output bottlenecks, message passing, interrupt latency, and housekeeping. The 'C40 reduces bottlenecks with dual identical buses and a six-channel DMA coprocessor for both internal and I/O operations.

- **LINPACK**

  The LINPACK benchmark, also known as Dongarra's benchmark, uses the Gaussian elimination process to solve a large matrix. A benchmark [10] for a 64-node hypercube, 1000 × 1000 matrix, three MFLOPS per node, 20-Mbytes/s links, and message overhead of 100 μs gives approximately 45% processor utilization. When this is extrapolated to the 'C40, where each node performance is 50 MFLOPS, 20 Mbytes/s link speed, and message overhead of approximately 100 cycles or 5.0 μs, a 64-node 'C40 hypercube can be expected to have a minimum of 45% or better processor utilization or a minimum LINPACK benchmark of 1.4 GFLOPS.

- **MIPS /MOPS** (Millions of Instructions/Operations per Second)

  These numbers are application-dependent because different instructions require different numbers of operations. For example, the 'C40 can perform as many as eight CPU and three DMA operations per cycle, which is equivalent to 275 MOPS.

- **Kernel tasks**

  Kernel tasks occur frequently and influence the system throughput. Benchmarking these kernels—for example, the Fast Fourier Transform (FFT) algorithm, FIR, matrix operation, and correlation—shows relative performance for different parallel computers.

- **Calculation Algorithm**

  The dart board algorithm [11], Figure 24, can be used to estimate the value of $\pi$.

### Figure 24.  Dart Board for $\pi$ Estimation
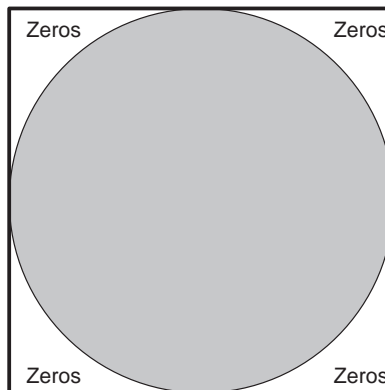
The dart board consists of a circle circumscribed by a square. The radius of the circle is one unit. Darts are thrown at the dart board in such a way that each dart has an equal probability of hitting any part of the square. If a dart lands within the circle, it is counted as 1; outside the circle, as a 0. At the end of the algorithm, the total number of ones are divided by the sum of the 1s and 0s. The estimate of $\pi$ is obtained as shown below.

$$\frac{\text{Area of circle}}{\text{Area of square}} \; = \; \frac{\pi \; (\text{radius})^2}{(2 \; \times \; \text{radius})^2} \; = \; \frac{\pi}{4} \qquad (6)$$

$$\frac{\text{\# of 1s}}{(\text{\# of 1s}) + (\text{\# of 0s})} \; = \; \frac{\pi \; (\text{radius})^2}{(2 \; \times \; \text{radius})^2} \; = \; \frac{\pi}{4} \qquad (7)$$

In the $\pi$ benchmark algorithm, each 'C40 node runs its own dart board algorithm for $\pi$ estimation. In a parallel system, the score increments in parallel on all nodes. At the end, the n estimates of $\pi$ are globally combined. Each 'C40 node generates a nonoverlapping set of random numbers.

True benchmarking can be done only by implementing the actual algorithm on all the systems and architecture configurations of interest. This is a demanding task because it requires programming resources and hardware cost.

## Algorithm Analysis

This section discusses algorithm load distribution and speedup for multiple node systems.

### Load Distribution

If all nodes are identical (memory size, communication channel, clock speed) and if they execute a program with equal data size, they will all complete execution at the same instant. In this case, the 'C40 computation time, $T_i$, is the same for all nodes. However, if data size is unequal or if the start of execution is staggered, $T_{max}$ is calculated as follows:

$$n = \text{number of 'C40 nodes; } T_i = \text{computation time at 'C40 node i;} \qquad (8)$$

$$T_{max} = \max\{T_1, \; T_2, \; ............ \; T_N\} \qquad (9)$$

$$T_{wasted} = \sum_{i=1}^{N}(T_{max} - T_i \;); \qquad (10)$$

$$V = \text{Variation ratio} = \frac{T_{max}}{T_{min}}, \; \text{where } T_{min} = \min\{T_1, \; T_2, \; ............ \; T_N\} \qquad (11)$$

For an ideal parallel processing system with a perfect load balancing, the variation ratio is:

$$V \cong 1 \qquad (12)$$

and

$$T_{wasted} = 0 \qquad (13)$$

### Speedup

Speedup and cost are two critical parameters for evaluating parallel systems against sequential systems. Speedup is defined as:

$$\text{Speedup} = \frac{\text{worst-case running time of fastest known sequential algorithm for problem}}{\text{worst-case running time of parallel algorithm}} \qquad (14)$$

It may appear that speedup is directly proportional to the number of 'C40 nodes. This may not always be true, especially when data dependencies or parallel processing system restrictions prevent dividing the task into equal parallel tasks. Assuming that a program could be subdivided into n equal subprograms for parallel implementation on n 'C40 nodes, the system cost is defined as:

$$\text{Cost} = \text{program execution time} \times \text{number of 'C40 nodes used} \tag{15}$$

$$c(n) = t(n) \times p(n) \tag{16}$$

If the parallel algorithm is perfectly load balanced, an optimum cost is obtained. If a parallel system is not load balanced, several 'C40 nodes may be idle and contribute to higher cost. A parallel processing system is cost-optimal if the cost of the parallel execution matches the known lower bound on the number of sequential operations. You can reduce the running time of a cost-optimal parallel algorithm by using more 'C40 nodes. You can also maintain the cost-optimal equation and use fewer 'C40 nodes at the price of higher running time. For example, in the *odd-even merging* algorithm, if the number of elements to be merged is equal to n, then the upper bound on a sequential computer is O(n) [order of n]. If the same merging is carried out on a parallel processing machine with n 'C40 nodes, the cost is:

$$\text{Cost} = t(n) \times p(n) \tag{17}$$

where *t(n)* and *p(n)* are defined above.

$$= [1 + \log n] \times [1 + n \log n] = O(n \log^2 n) \tag{18}$$

(read as order of n log square n)

Parallel implementation of merging is not cost-optimal, because $O(n) < O(n \log^2 n)$.

Efficiency ($\eta$) of a parallel algorithm is evaluated as shown below.

$$\eta = \frac{\text{worst-case running time of fastest known sequential algorithm for problem}}{\text{Cost} = p(n) \times t(n)} \tag{19}$$

For example, if a program is executed sequentially in 100 ms and the same program is executed parallel over 10 processors in 20 ms, efficiency ($\eta$) is 50%.

As the value of n increases, other factors that must be evaluated are the cost of capital investment, maintenance, programming, etc.

## Data Decomposition

Data decomposition is an effective tool in evaluating architecture performance. A brief review of different techniques for data decomposition and task allocation is presented in this section.

### *Uniform Partitioning*

The task execution is independent of input data. Data is completely deterministic. In parallel execution, the data and computation are equally distributed among all 'C40 nodes for best load balancing and maximum efficiency. Uniform partitioning is used in convolution, FFT, and filtering algorithms.

### *Static Partitioning*

In Hough transformation, computation is proportional to the number of edges present and not to the size of the image. For example, if only a chair is present in the right-hand corner of a $512 \times 512$ image, then the 'C40 nodes with that portion of the image will be busy computing edges, whereas the rest of the 'C40 nodes will be idle because no edges are present. This load is not balanced, and the data is not decomposed. Static load balancing requires a priori information as an input for data decomposition.

### Weighted Static

Computation is a function not only of granule size and data density but also of spatial relationship of significant data. For this type of data, a weighted static scheme is useful for load balancing and data decomposition. In Hough transformation, information regarding granular size and the number of edges can be easily obtained. Two granular sizes may have equal amounts of significant data (edges), but in one case, the edges may not be related to each other at all. In the other cases, the edges may be spatially connected to form a certain object, and a large amount of computation may be required to identify an object.

### Dynamic

Uniform, static, and weighted static schemes decompose data after the data characteristics are known. However, it is possible to assign new tasks to a particular 'C40 node in advance by predicting that the current task will be completed after a certain period.

Table 2 shows [13] execution time for an eight-node system implementing a motion-detection algorithm. The dynamic method yields near perfect data decomposition (close to 1 variation ratio). This is also true for other algorithms that have data dependency, spatial relationship, and data variation across the inputs.

**Table 2. Comparative Time for Four Methods to Execute an Algorithm**

| Total Eight Nodes | Uniform | Static | Weighted Static | Dynamic |
|---|---|---|---|---|
| T maximum | 3500 | 1100 | 1200 | 1000 |
| T minimum | 100 | 200 | 750 | 950 |
| Variation Ratio V | 35 | 5.5 | 1.6 | 1.1 |

## Conclusion

The TMS320C40 is a nearly perfect node for implementing hypercube, ring, mesh, and pyramid architectural configurations. For integrating a parallel processing system, a 500-MFLOPS system can cost as little as $20,000 to $30,000. Many programming and design tools are now available for the 'C40, and many more will be available soon. For quick time to market, many board-level 'C40-based products that are available now can be integrated. Many low-cost and low-power derivatives of this processor from Texas Instruments are expected. Computationally intensive algorithms will be ported to the 'C40 and will be available as off-the-shelf algorithms. The TMS320C40 is destined to become the most commonly used parallel-processing node.

## Definitions

'C40 node:      A node that consists of a processor, execution units, memory modules, communication port, and other necessary hardware.

Dimension of hypercube structure:      $n = \log_2 p$ (p = number of processors)

Distance:      The number of links used by the message to travel from the source 'C40 node A to the destination 'C40 node B.

Link:      A physical connection between any two nodes.

| | |
|---|---|
| Hamming Distance (HD): | The number of bits that are different in two A and B binary addresses. For example, if A is (1010101) and B is (0101010), then the Hamming distance is 7. |
| Node/Vertex (N): | A processing element that is in the network. Also, the point in the graph where one or more edges meet. |
| Parallel Paths: | Paths with equal lengths and nonoverlapping edges. |
| Computation time $(T_i)$ | Computational time on a single-processor system. |

Maximum computation time $(T_{max})$

$$T_{max} = \max \{T_1, \ T_2, \ ... \ T_N\}$$

Minimum computation time $(T_{min})$

$$T_{min} = \min \{T_1, \ T_2, \ ... \ T_N\}$$
where $N = P = \#$ of processors

Variation Ratio (V)

$$V = \frac{T_{max}}{T_{min}}$$

Wasted time $(T_{wasted})$

$$T_{wasted} = \sum_{i=1}^{N}(T_{max} - T_i \ );$$

| | |
|---|---|
| 'C40 | Texas Instruments TMS320C40 floating-point DSP. |
| Efficiency | $E_p = S_p/p$ with values between $(0,1)$, is a measure of processor utilization in terms of cost efficiency. |
| Speed-up $(S_p)$ | $S_p = T_s/T_p$, where $T_s$ is the algorithm execution time when the algorithm is completed sequentially; whereas, $T_p$ is the algorithm execution time using p processors. |
| Parity: | The number of 1s in a node's binary address. Parity of a node is positive if there are an even number of 1s in the node's binary address; if the number of 1s is odd, parity is negative. |
| Path: | The ordered list of processors visited by the message in going from the source 'C40 node A to the destination 'C40 node B. |

# References

[1] Tim Studt. R&D Magazine, July 1992.

[2] TMS320C4x User's Guide, Dallas; Texas Instruments Incorporated, 1991.

[3] Parallel 2-D FFT Implementation With TMS320C4x DSPs Application Report, Dallas; Texas Instruments Incorporated, 1991.

[4] K. Hwang and D. DeGroot. Parallel Processing for Supercomputers and Artificial Intelligence. New York: McGraw-Hill, 1989.

[5] R.H. Kuhn and D.A. Padua. Tutorial on Parallel Processing. IEEE Computer Society, 1981.

[6] Z. Hussain. Digital Image Processing. Ellis Horwood, 1991.

[7] S.G. Akl. The Design and Analysis of Parallel Algorithms. Englewood Cliffs: Prentice-Hall, Inc., 1989.

[8] Y. Saad and M.H. Schultz. Topological Properties of Hypercubes.

[9] Product Catalog, Sonitech International, Wellesley, Massachusetts 1992.

[10] Hungwen Li and Q. F. Stout. Reconfigurable Massively Parallel Computers. Englewood Cliffs: Prentice-Hall, 1991.

[11] G. Fox & others. Solving Problems on Concurrent Processors, Volume I. Englewood Cliffs: Prentice-Hall, 1988.

[12] M. Ben-Ari. Principles of Concurrent and Distributed Programming. Englewood Cliffs: Prentice-Hall, 1990.

[13] A.N. Choudhary and J.H. Patel. Load Balancing and Task Decomposition Techniques for Parallel Implementation of Integrated Vision Systems Algorithms, Proceedings of Supercomputing 1989, November 13–17, 1989, pp. 266–275. New York: IEEE Computer Society.

[14] F. Thomson Leighton. Introduction to Parallel Algorithms and Architectures. San Mateo: Morgan Kaufmann Publishers, 1992.

[15] Sonitech internal program "Iterative Method for 'C40 Communication Port Interconnection."