Presentation of the Cubi9000: A Graphics System based on Inmos T800 Transputers

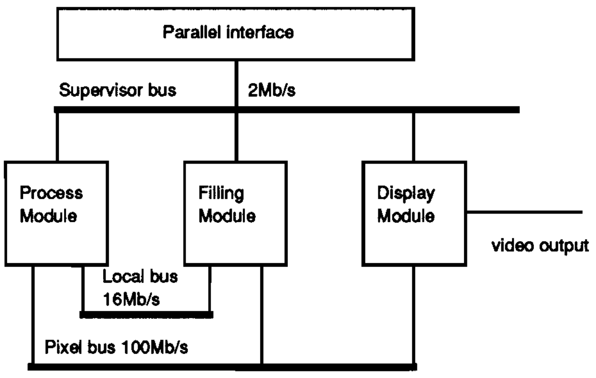
France Glemot

Caption[1], Rennes, France

1. Introduction

The Cubi9000 family includes a range of products from the 3D graphics terminal up to the 3D graphics workstation. The Cubi9000 when configured as a 3D graphics terminal connects to a host computer via a parallel interface from Digital Equipment Corporation. The Cubi9000 configured as workstation includes a UNIX[2] based mini-computer with a 32 bit microprocessor, mass memory, device handlers (mouse, tablets, encoders) and communication drivers to external systems.

2. General hardware overview



The basic hardware comprises three modules communicating via three different buses.

[1]The company Caption is a subsidiary of the company Telmat which manufactures telecommunication systems and minicomputers. Caption specialises in 3D image synthesis, it develops, builds and sells 3D graphics systems.
[2]Unix is a trademark of ATT Bell Laboratories

2.1 Display module

The Display module includes a frame buffer and a video interface:

The frame buffer comprises 40 1024*1024 bit-planes with a 16 bit hardwired Z-buffer providing 16 million simultaneous colours and uses a conditional writing-by-depth test for hidden part removal. The image formats may be dynamically programmable from 512*512 to 1024*1024 pixels.

The video interface provides an interlaced or non-interlaced video output, and an external synchronisation (genlock) is available for the video formats. It is possible to mix two plane images in depth at the video pixel rate, before the input to the digital to analog converters. Furthermore one of these planes can receive video images via a digitizer entry with blue box capabilities. So real and synthetic images can be mixed in real time.

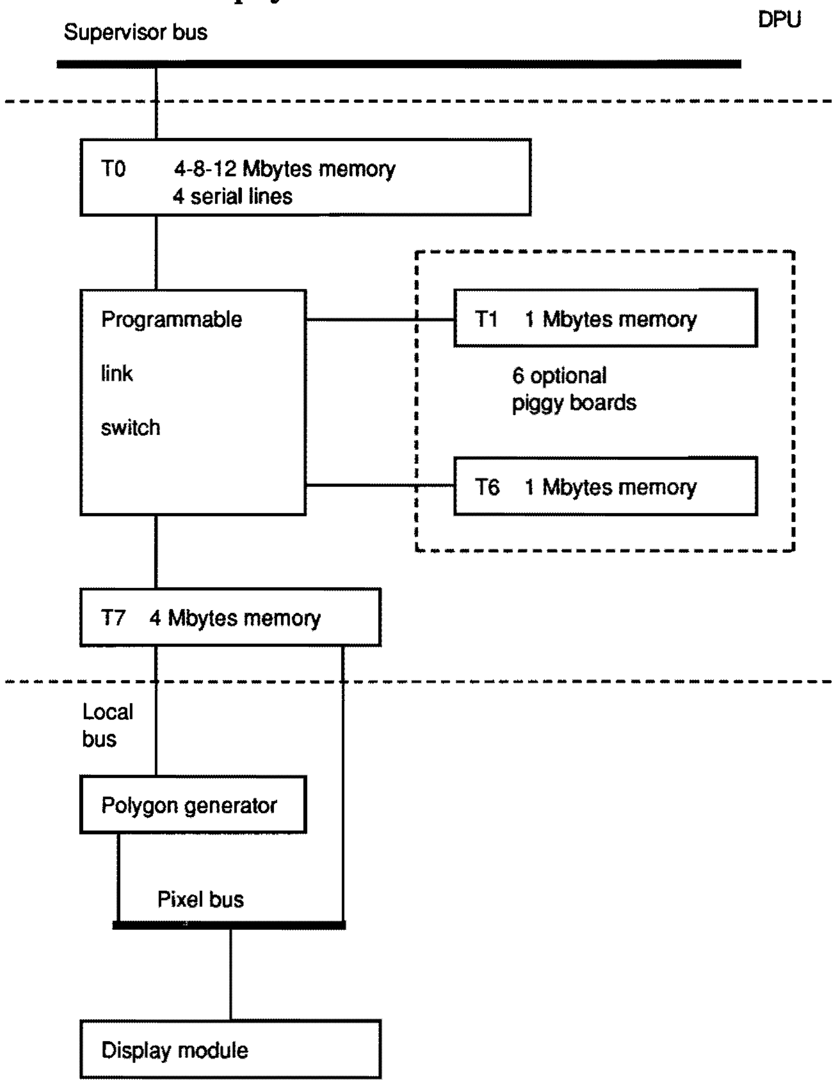
2. 2 Filling module

The filling module comprises one or two polygon generators performing polygon shading (Gouraud shading at 20 Mpixels/s per generator). Each polygon generator consists of 2 boards, the first one computes the interpolation of colour and depth along the edges and the second computes the interpolation of colour and depth of the horizontal lines filling the polygons. These two boards are completely hardwired and offer 120 Mips.

2.3 Process Module

The process module comprises one or two Display Processor Units (DPU). The DPU is a parallel work flow board with several INMOS transputers, the number of on-board processors depends upon the processing power required. This module contains the display list and handles the image generation algorithms.

3. Architecture of the Display Processor Unit



Block diagram of the Display Processor Unit associated with the Polygon Generator and the Display Module

3.1 Description of the transputer T0

The transputer T0 is non-optional, its hardware resources are the following:

- 4, 8 or 12 Mbytes dynamic RAM memory accessible by the Host via the Supervisor bus. This bus is shared between the Host and T0

- access to all other boards, via the Supervisor bus, in order to configure the Cubi9000

- four RS232 serial lines for connecting graphics peripherals

- two synchronization signals: an interrupt signal to the host and a video top-of-frame signal from the video board of the display module

- four links: one dedicated to the host (for program loading and urgent messages), the other three being connected to the programmable link switch (for communication with the other transputers).

The main tasks of T0 are: decoding of the received messages from the host, execution of the corresponding function, management of the graphics peripherals, sending back information to the host, processing geometric transformations (for Gouraud shading) and, if the DPU has only 2 transputers T0 and T7, it is responsible for high quality rendering effects (Phong shading, texture mapping, solid textures...)

3.2 Description of the transputer T7

The transputer T7 is non-optional, its hardware resources are the following:

- 4 Mbytes dynamic RAM memory (not shared and not accessible from the Supervisor bus, the Local bus or the Pixel bus)

- access to the Local bus in order to send data to the polygon generators

- access to the Pixel bus in order to write/read pixels directly in/from the frame buffer of the display module

- four links connected to the programmable link switch, for communication with the other transputers.

The tasks of T7 are mainly to collect data processed by the other transputers in order to send them to the polygon generator (for Gouraud shading) or to the frame buffer (for Phong shading). It performs anti-aliasing post-processing, using a choice of two techniques: oversampling and sub-precision for pixels belonging to the edges. It computes the features of polygons projected on the screen space (for Gouraud shading) and, if the DPU has only two transputers (T0 and T7), it produces high quality rendering effects (Phong shading, texture mapping, solid textures...)

3.3 Description of the transputers T1-T6

The transputers T1 to T6 are optional. They are equipped with 1 Mbytes dynamic RAM memory which is not shared and cannot be accessed by other processors. They have 4 links connected to the programmable link switch for communication with the other transputers.

The tasks of T1 to T6 are mainly: processing geometric transformations, computing the features of polygons projected on the screen space, processing high quality rendering effects (Phong shading, texture mapping, solid textures...).

The six optional piggy boards allow the processing power to be increased from 20 Mips/3 Mflops to 80 Mips/12 Mflops per DPU.

The reset, analyse and error signals of each transputer in the network are linked together by a daisy chain mechanism.

3.4 Description of the programmable link switch

This programmable link switch allows any link from one transputer to be connected to any link of any other one (link0 from T0 is not available for this purpose). The 8 transputers are linked together via this programmable link switch, allowing an optimum configuration to be provided for each application. Thus the Cubi9000 is actually a multi-mode device, able to support efficiently several image synthesis computing techniques (from Z-buffer to ray tracing and from polygon to octree modeling).

4. Software functionalities

The graphics library is written in the C language, but the calling level may support other languages such as Fortran, Pascal, ADA.

The graphics library comprises four independent parts: initialisation and configuration of the Cubi9000, management of the display list, management of the graphics peripherals and 2D functions.

4.1 Initialisation and configuration of the Cubi9000

The functionalities are the following: open/close a session without modification to the configuration or with default values, load colour look-up tables, select the video format, parameters of the display monitor, colour and depth of the background, erase/display a plan of the frame buffer, select the visualization mode (real time or not, mixing of 2 planes of the frame buffer or not), digitise a static image (colour slide, photography...) or dynamic images (camera, magnetoscope...) for the whole of an image or just those parts of it detected by a blue box.

4.2 Management of the display list

The display list comprises of 5 components: the computing features of the scene, the cameras, the objects, the light sources, the data tables (reflectance, texture...).

An entire component of the display list may be: added, deleted, duplicated, or substituted. A field of a component may be set or get.

4.3 Computing features of the scene

These computing features are the following:

- computing mode (normal, designation, picking)
- aliasing mode (none, oversampling, sub-precision)
- geometric description level (bounding box, object)
- rendering level (wire frame, Lambert, Gouraud or Phong shading, specific effects).

Cameras

The cameras are defined by:

- projection (orthogonal, conic_z, conic_1/z, conic_a+b/z)
- position of the observer
- features of the viewport (window on the screen)
- features of the window of projection
- features of the mask on the window of projection.

Objects

The objects are defined by:

- global attributes:
    - drawing flag (define whether or not the object must be taken into account)
    - bounding box features
    - position and coefficient of reduction/expansion
    - colour flags and colour coefficients
    - smoothing flags and smoothing coefficients (colour, geometric and arbitrary smoothing)
    - aspect flags and aspect coefficients (coefficient for Phong shading: - diffuse, specular, brilliance, transparency: min, max, thickness, and reflectance)
    - flags for the geometric description and rendering levels
    - flags for the processing of backfaced polygons (normal, interior, or sections which give different views of an object modified by a front clipping plane, according to a normal inside-outside description or outside only description
- polygon description table
- vertices table

- polygon colour table
- aspect table.

Light sources

An ambient light source will always exist. The light sources are defined by type (sun, point-source, spot), the intensity, the position (suns are located at infinity), the direction (not relevant for the point-source), the colour, and the cone angle (relevant for spot only).

Data tables

Two tables may be used: the reflectance table and the texture table.

4.3 Management of the graphics peripherals

Four RS232 lines on the DPU allow it to be connected to: an alphanumeric terminal, a 2D tablet, an 8 encoders board, a magnetoscope (format BVU, BETACAM, 1 inch), and to any other peripheral programmable via an RS232 line.

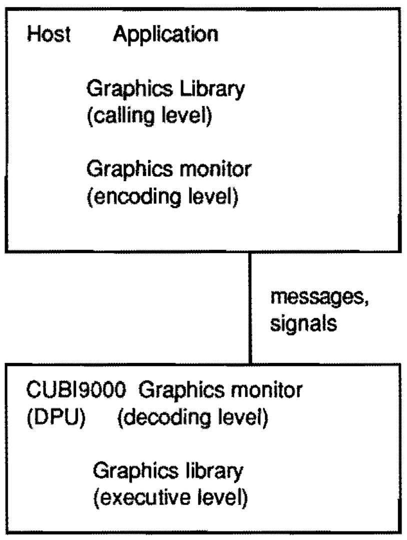
The reading modes are compatible with those of Phigs: Sample, Request, Event. The data issued from a graphics peripheral (other than a magnetoscope) may be sent directly to the host, processed on the DPU and sent to the host, or processed and used in order to modify a field of a component of the display list between two image computations. This is done by the DPU, the host has just to send the command for image computations.

The alphanumeric terminal may be initialized, activated/stopped, read and written. The 2D tablet may be initialized, activated/stopped, configured (operating mode, resolution, sample speed, origin, size..) and read. The 8 encoders may be initialized, activated/stopped, configured (sample speed, minimum threshold) and read. The magnetoscope may be initialized or instructed (play, stop, forward, rewind, read/write/go_to a time code, record, standby). An interface between the magnetoscope and the Cubi9000 based on PC computer is necessary.

4.4 2D functions

All the components of a pixel R, G, B, and Z or only one of them may be read, written or written conditionally upon its Z value, in a rectangular space of the frame buffer. Cursors and reticles may be defined, selected, positioned, moved, erased. Coloured vectors and polygons may be drawn in any plane of the frame buffer. A character font may be defined, loaded, selected. By default a standard font is defined, loaded and selected. A string may be written in any plane of the frame buffer using the following features: colour, position, direction, thickness and reduction/expansion for character height, width, spacing.

5. Software architecture


Host
Application
Graphics Library (calling level)
Graphics monitor (encoding level)
messages, signals
CUBI9000 (DPU)
Graphics monitor (decoding level)
Graphics library (executive level)

The communication between the host and the Cubi9000 is based on messages and synchronization signals. The application is located on the host for the obvious reason of portability, but some parts of it could be exported in order to complement the library, even for non graphics processing. The calling level of the graphics library must be located within the application, it performs some coherence checking on the parameters. The executive level must be efficient, and for this reason it is implemented in firmware.

The graphics monitor has to transfer the graphics commands from the host to the DPU, to synchronize them and to transfer data from the DPU to the host.

The urgent messages are sent to the address of link0 of T0, they are scanned automatically and processed by a concurrent process. The mechanism of urgency is used when an error occurs or when the user wishes to terminate immediately an animation sequence. The mechanism of urgency stops the execution of the current command and re-initialize some variables which are shared between the host and the DPU.

Another concurrent process manages the graphics peripherals. The tasks of the encoding level of the graphics monitor are to collect commands and their associated parameters from the library and packet them up in a format understood by the DPU. It is required to wait if the command input fifo on the DPU is full and otherwise send the packet to that fifo. If a reply is required, it waits for the end of the execution and, in which case, gets data from an output data buffer on the DPU, it unpackets them into a format understood by the host and stores them in the place defined by parameters.

The tasks of the decoding level of the graphics monitor are to wait if the command input fifo is empty, decode the packet, performing the corresponding function or using the executive level of the graphics library to send it to the underlying network. Finally, if a response is required, the data is put into the output data buffer and then a synchronisation signal is sent to indicate to the host that the execution of the command is finished.

The command input fifo and data output buffer are shared, at the graphics monitor level, by both the host and the DPU (T0). Conflicting accesses are resolved mainly by the rule that the variables shared may be read by both, but must be written by only one. For instance, the host is not allowed to reset the status of the DPU itself, and has to send a urgent message to the DPU.

When a command does not need a synchronization signal at end of its execution, both the host and the Cubi9000 can work in parallel. Furthermore the command input fifo serves to regulate the flow of commands between the host and the Cubi9000.

6. Constructing load balanced systems

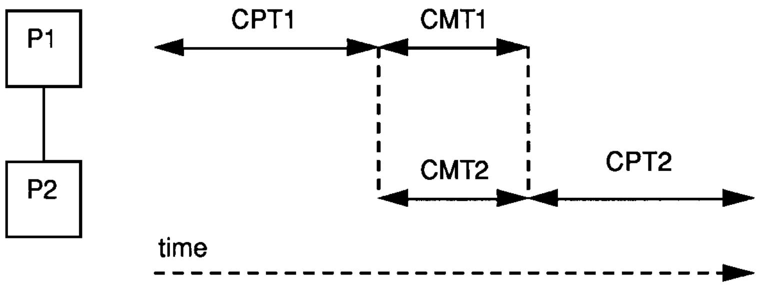
6. 1 Some general rules

To achieve good performance, a program may be split into several processes implemented on a network of transputers. Communications within such a network use the links as channels.

6.1.1 Simple architecture

Consider first a case in which there are two transputers running processes P1 and P2, if one process is (much) faster than the other, one of the processors will be under utilised, but we will now assume the system is correctly load balanced. The compute times and communication periods are denoted, respectively, CPT and CMT.

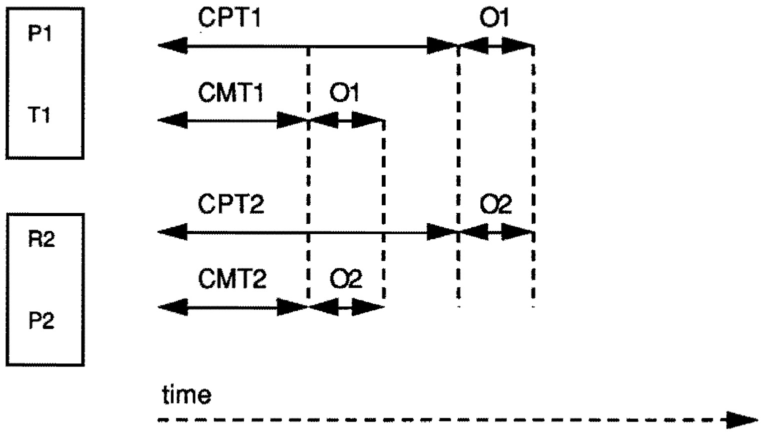
Load balanced systems imply CPT1 = CPT2 (to a close approximation) and synchronized communication implies CMT1 = CMT2.



The data processing rate, to an external observer, is CPT1 + CMT1, for a non-balanced system we have Max(CPT1, CPT2) + CMT1.

6.1.2 An improved architecture

We now consider the same problem, but with two additional processes: a transmitter process T1 on the first transputer and a receiver process R2 on the second transputer. T1 and P1 operate in parallel, communicating via two buffers and using a flip-flop mechanism between the two buffers. A similar situation exists for R2 and P2 on the second transputer. We may consider that T1 and R2 introduce a small offset in the execution of P1 and P2 because each of them uses a DMA independent of the ALU, the offset is due to the fact that the internal bus must be shared between the DMA and the ALU and the fact that a time slicing is necessary between P1 and T1 (and between P2 and R2). The effect of time slicing is, however, very small. We denote these offsets as O1 and O2, and assume that any difference between O1 and O2 is small.



The perceived external data processing rate is CPT1 + O1, and for a non-balanced system we have Max(CPT1+O1, CPT2+O2).

The results are better than those obtained above and for a longer pipeline we can generalize in the following way:

> On the first transputer, the processes are P1 and T1 (and maybe an R1 for the external world); on the following transputers, the processes are Ri, Pi and Ti; and on the last transputer, the processes Rn and Pn (and maybe a Tn for the external world).

Ri and Ti are secondary processes which arise due to this manner of implementation, and only the Pi processes perform useful work for the application. This is the reason why the processes Pi need not wait for the communication processes Ti and Ri.

We have CPTi $\geq$ CMTi for Ti and the same for Ri.

6.1.3 Advantages of the pipeline architecture

Splitting the program into several processes results in each of them being shorter and thus they may be held within the internal memory of the transputer. Generally this also applies to the data. The internal memory accesses are much faster than external memory accesses (by a factor of 2 or 3 times). The dispatching of commands and data and the collection of results are simpler than within a parallel architecture.

6.1.4 Advantages of the parallel architecture

The parallel architecture does not suffer the time delays of a pipeline architecture. incurred when the pipeline is being loaded and emptied. The parallel architecture is more efficient than a pipeline architecture because it uses fewer processes , implying less time slicing and fewer data transfers. If splitting the code results a large amount of data to transfer then the rule that a computing process must wait for communication may not be hold.

6.1.5 Advantages of the mixed (pipeline and parallel) architecture

In practice, transputer based systems may have a mixed architecture in order to communicate easily with the external world and to achieve efficiency. Nevertheless with the same number of transputers several architectures are possible, as is shown below by considering three application examples.

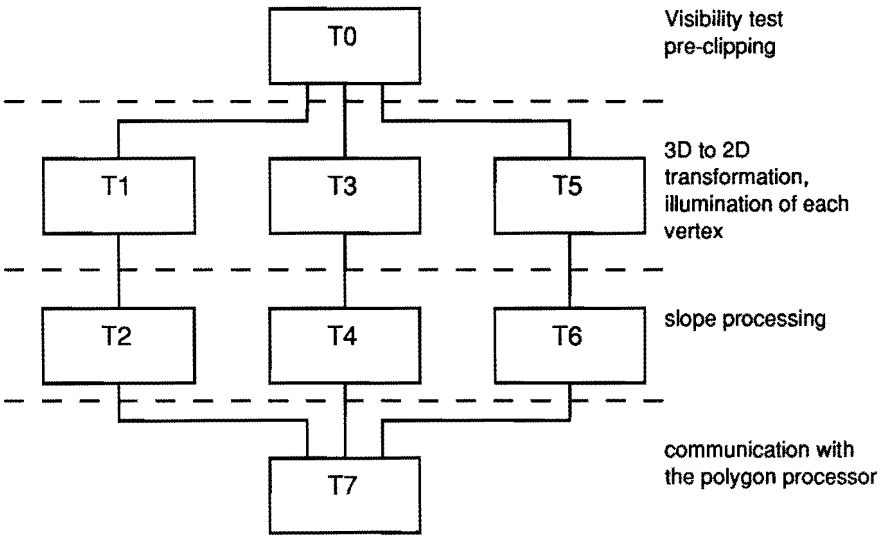
6.2 Three examples of applications

6.2.1 Real time simulator

In this case the DPU performs the 3D to 2D transformations and determines the edge slopes for each polygon; the polygon generator being responsible for filling these polygons.

There are three steps in the basic algorithm: the visibility test (backface removal) and pre-clipping, the 3D to 2D transformation with clipping and illumination of each vertex, and the slope processing (in which the gradient values of X,Y,Z,R,G,B are determined). The results are communicated to the polygon generator.

The resulting transputer network architecture is organized in three pipelines according to the following two rules: the work load for T1 to T6 must be similar, and the work load for T0 and T7 must be less than 33% of the work load of others transputers.


T0
Visibility test pre-clipping
T1
T3
T5
3D to 2D transformation, illumination of each vertex
T2
T4
T6
slope processing
T7
communication with the polygon processor

This logical architecture is similar to the physical architecture, each transputer having a receiver process R, a transmitter process T and one or several computation process(es) P.

The Cubi9000 can process and display from 4000 to 40000 3D Gouraud shaded polygons per second (with an average of 300 pixels/polygon).

6.2.2 High quality images

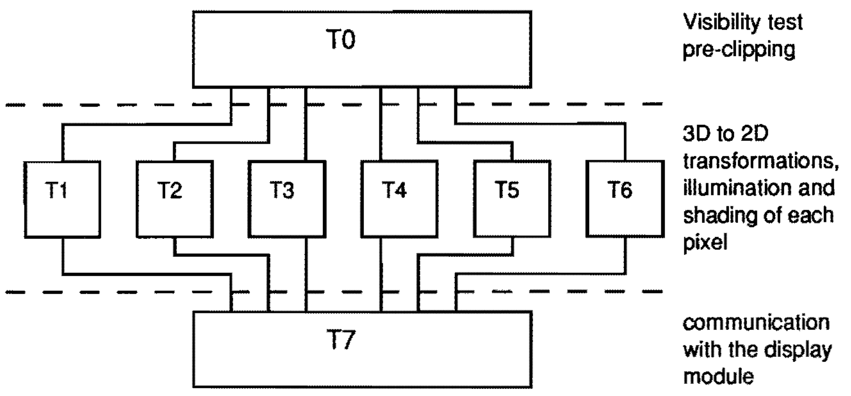
In this case the polygon generator cannot be used, and the DPU has to compute each pixel value, so the program is larger than that for Gouraud shading. Furthermore there are additional specific effects which require large data tables (reflectance, texture...), with the result that the internal memory of the transputers is too small to hold the program and data.

The amount of computing power required to compute illumination and shading at each pixel is much greater than the computing power needed for geometric transformation, with the consequence that the process cannot be split at this level. It is not possible to make a split between illumination and shading, without having to transfer a large amount of data, and incurring large communication overheads.

Furthermore in the architecture shown below, a single source program can be used with 2, 4, 6 and 8 transputers on the DPU.

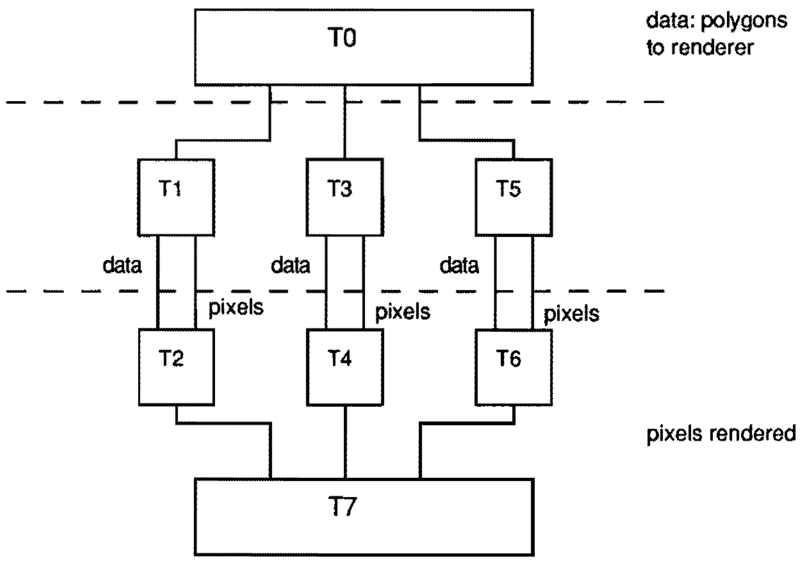
The basic algorithm is decomposed in three steps: the visibility test and pre-clipping, the 3D to 2D transformations with clipping, illumination and shading of each pixel, and the communication with the display module.

Using the rules previously defined, the logical network architecture is:


T0
T1
T2
T3
T4
T5
T6
T7
Visibility test pre-clipping
3D to 2D transformations, illumination and shading of each pixel
communication with the display module

This diagram represents only a logical architecture, in practice T0 will have only 3 links connected to the programmable link switch. So there is a by-pass mechanism on T1, T2 and T3 in order to dispatch data to the computing process within the transputer or to the receiver process on the transputers T4, T5 and T6. On T4, T5 and T6 another by-pass is used to transmit processed pixels received from T1, T2 and T3 directly to T7 or to transmit the data to the computing process within the transputer. Each transputer has a receiver process R, a transmitter process T and one or several computation process(es) P.

The physical architecture is the following:



The Cubi9000 can process and display from 3500 to 40000 Phong shaded pixels.

6.2.3 Octree

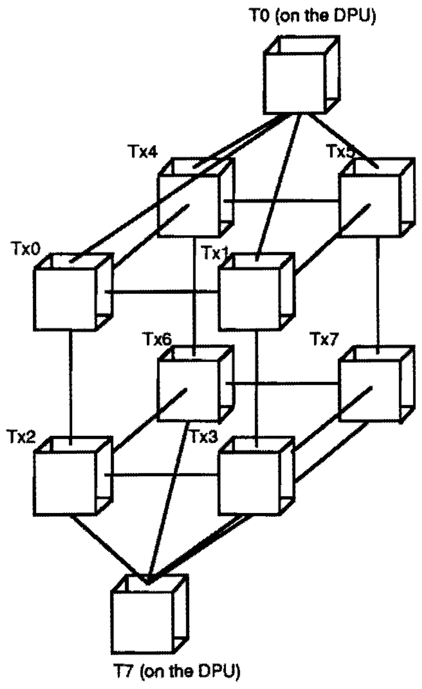
For octree applications another development based on a further board is now considered. This board is based on two modules of 4 transputers each, and connected via 7 links to a DPU equipped with the two transputers T0 and T7 (note that link0 of T0 is not available for such a purpose).

The architecture chosen is based on a cube with 8 transputers, one at each vertex. The advantages of such an architecture is that it is similar to the octree organization which makes the data dispatching algorithm easier to implement, and there is a smaller number of connections between the transputers within the cube thereby reducing the transfer time for data.

It will be observed that most operations needed involve modifications to the octree dispatching because they alter the tree organization. Such operations are the creation of the tree, creation of objects (prosthesis...), segmentation (detection of structures corresponding to an organ, tissue...), rotation or translation of the tree (modification of the reference).

The only difference between the logical architecture and the physical architecture is the link between the transputer T0 on the DPU and the transputer T0 on the other board. There is a logical link from T0 on the DPU and T0 from the other board, but physically this link uses T1 (on the other board).

The logical architecture is the following:

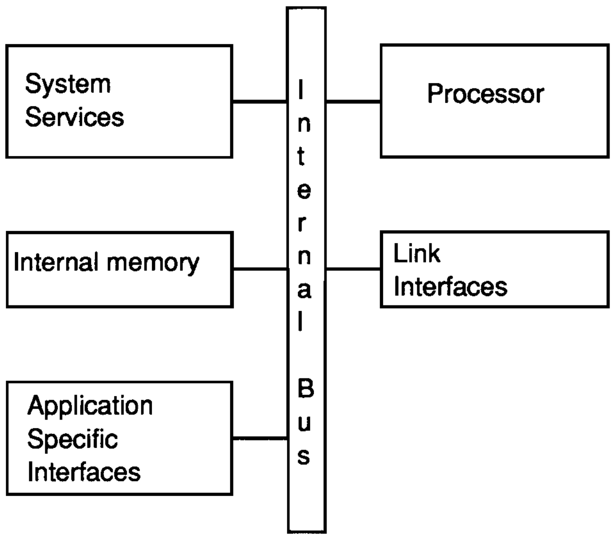


7. The transputer and Occam[3]

7.1 Introduction

This section presents an overview of the transputer family and a description of the main concepts in Occam with their associated implementation on transputers.

7.2 Overview of the transputer family



7.2.1 System Service

The main signals are the following:

- reset: this input signal resets the transputer. On the T414 and T800, this signal also resets the configuration of the external memory interface (code and data located on external memory are destroyed)

- analyse: this input signal resets the transputer prior to debugging. On the T414 and T800, the configuration of the external memory interface, code and data located on external dynamic RAM memory are not changed.

- error : this output signal is an indication to the external world when an error occurs at execution time. In the T414 and T800 an additional error

[3]Occam and INMOS are trademarks of INMOS

flag is internally set for debugging facilities. Error checking possibilities are, for instance, array boundary overflow, and arithmetic overflow.

- boot: this input signal permits booting from a external ROM memory. Another possibility is to boot via a link.

7.2.2 Internal Memory

The on-chip memory is 2 Kbytes in the T414 and 4 Kbytes in the T800. No caching facilities are provided. The accesses to code and data located in this memory are much faster than to those located in the external memory. The programmer may force certain code and data to reside in a specific address area, for instance in this internal memory.

7.2.3 Application Specific Interfaces

In order to allow the design of an interface with a minimum of external components, the transputers may include a controller for SCSI disk, graphics or memory. Timings and standard control signals are programmable by software.

7.2.4 Internal Bus

This bus multiplexes 32 bit addresses and 32 bit data, in both the T414 and the T800,

7.2.5 Processor

The T414 has an integral 32 bit integer Arithmetic and Logic Unit (ALU). The T800 has a 32 bit integer ALU and a 64 bit IEEE floating point ALU. The 30 MHz T800 offers 10 Mips and 1.5 Mflops.

7.2.6 Link Interfaces

A serial input line and a serial output line are associated with each link interface. The T414 and the T800 provide 4 link interfaces. A link is a point to point connection from a transputer family chip to a transputer family chip. Within a network, transputers are connected via links.

Point to point connection links have many advantages over multiprocessor buses:
- no contention for the communication mechanism,
- no capacitive load penalty as transputers are added to a system,
- no saturation of the communication bandwidth, and furthermore an higher, total communication bandwidth in the system.

These advantages are independent of the number of transputers in the system.

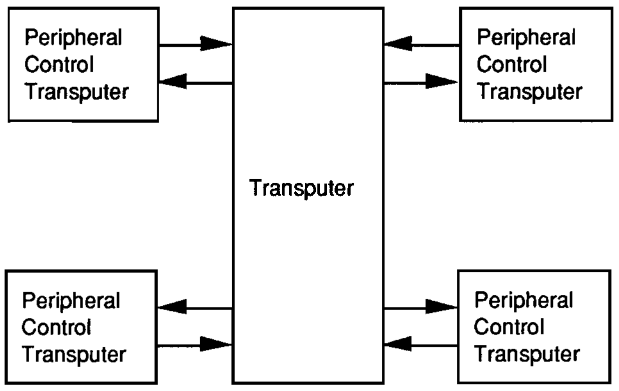
On the T414 and T800, the link interfaces incorporates two separate DMA devices, one for each direction. Thus communication and processing can take place independently, with only the internal bus being shared.

Link adaptors are part of the transputer family, they allow transputer links to be interfaced to a non-transputer sub-system. They convert data to and from a link to an 8 bit parallel port, taking into account the specific communication protocol of the links. Link adaptors may have two operating modes depending on whether they work like two uni-directional parallel ports or single bi-directional parallel port. The features and advantages of link adaptors are the following: data reception is asynchronous, which implies that communication is independent of the clock phase and transfers can take place in both directions simultaneously (but with some speed penalty).

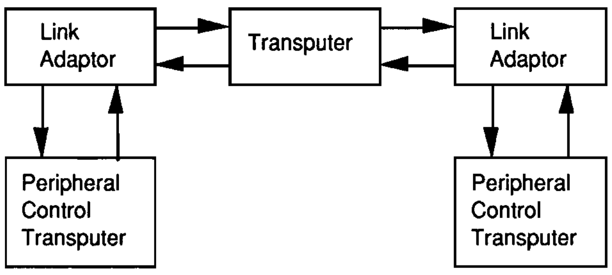
7.2.7 Peripheral Interfacing

Three methods may be considered:

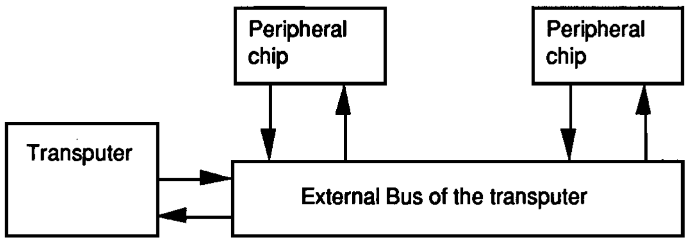
- a transputer with 1,2,3 or 4 peripheral control transputers
- a transputer with 2 link adaptors
- a transputer with peripheral addresses space mapped in the memory space.


Peripheral Control Transputer
Peripheral Control Transputer
Transputer
Peripheral Control Transputer
Peripheral Control Transputer

a transputer associated with 4 peripheral control transputers



a transputer associated with 2 link adaptors



a transputer with peripheral addresses space mapped in the memory space (the peripheral is controlled by memory accesses issued as a result of a port inputs and port output, cf next paragraph Concepts in Occam)

7.3 Concepts in Occam

Occam is a programming language for the transputers, it is a high level language giving a maximum program efficiency and is able to exploit the special features of the transputers. Occam and the transputer were designed together. Occam can be used as an harness to link modules written in standard languages (C, Fortran, Pascal...). This harness includes all information concerning the configuration.

Occam provides a framework for designing concurrent systems using transputers.

7.3.1 Processes

The software building block in Occam is called process. A system is an interconnected set of processes communicating via point to point channels. A hierarchically structured system is a set of processes designed themselves as a set of (sub)processes. Each process is an independent unit of design, completely specified by the messages it sends and receives. A process starts, performs a number of actions and then terminates. An action may be a set of any number of sequential or parallel processes. Procedures and functions are processes.

When two processes of different priority are concurrent, the process with the larger priority must stop (because it is waiting for a communication for instance) or complete, before than the other may run.

Concurrent processes with low priority are periodically timesliced (generally the running period is some hundreds of microseconds).

7.3.2 Channels

Channels are the logical medium used by parallel processes in order to communicate values, no variables are shared between such parallel processes. For each communication a channel, a transmitter process and a receiver process must be defined; the communication occurs only when the both processes are ready.

When two processes communicating via a channel are implemented on the same transputer, only data moves within the memory are performed.

When the channel is implemented on a transputer link, a protocol is used during the communication, its features and advantages are the following:

- a byte acknowledge, which provides reliable communication between a slow and a fast transputer and word length independency
- an anticipated acknowledge, which implies a continuous data flow
- a standard communication frequency of 10 Mhz for all the transputer family (nevertheless other frequencies exist: 5 Mhz, 20 Mhz..)
- no clock phase dependency, which imply that communications may happen between two independently clocked systems.

Communication via channels is synchronized automatically, no additional explicit programming is necessary. Furthermore the structure of a program which communicates between two processes is independent of whether the processes are executed on the same transputer or on two different transputers.

Communication via any channel may occur concurrently with communication via other channels and with program execution.

If the program is written in a language other than Occam, then a run-time system is provided which provides input/output to Occam channels (and which conforms to the link protocol if links are used).

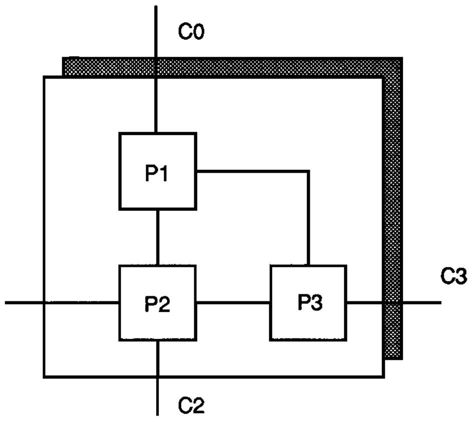
Peripheral access may use ports, which are extensions of the channel mechanism, for a block of information consecutively stored in memory. Nevertheless there is no synchronization mechanism associated with a port input and output. So a value read by a port input may depend upon the time at which the input was executed, and inputting at an invalid time would produce unusable data.

7.4 Implementation of the Occam on the transputer

The transputer and Occam were designed together. Hardware features are provided to support the special features of Occam. Whatever the number and kind of transputers used, the Occam level interface remains the standard interface.

A transputer link interface may be used to implement two channels, one for each direction. A program running on a transputer is called a process. All transputers incorporate a timer providing a clock which can be used by any number of concurrent processes for internal measurement or for real time scheduling. The hardware interface between a dedicated sub-system and a transputer based system has to use one or more links. During the development stage, the sub-system could easily be simulated by an Occam implementation of the associated process.

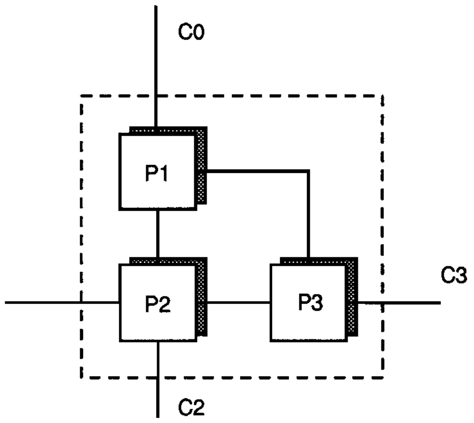
A system can be built with a single transputer independently of the number of processes, or with a network consisting of a large number of transputers without any topological restrictions, excepting that the T414 and T800 transputers have only four link interfaces. In both cases the identical source program can be used. So that an Occam program may run on a single transputer when the cost must be optimised or on a network of transputers for performance optimisation.



Optimisation for cost:

A program on a single transputer with 3 processes P1, P2 and P3, 4 links to the external world, and 3 internal channels between the processes (data moves within the memory)

The transputer shares its time between the 3 processes



Optimisation for performance:

A program on 3 transputers with 1 transputer for each process, 4 links to the external world, and 3 links for communication between the transputers.

Each transputer executes a single process