IGNOU STUDY HELPER
SUNIL POONIA

Course Code : MCSE-011
Course Title : Parallel Computing
Assignment Number : MCA (V)/E011/Assignment/2018-19
Maximum Marks : 100
Weightage : 25%
Last Dates for Submission : 15th October, 2018 (For July session)
15th April, 2019 (For January session)

Question 1: Compare the advantages and disadvantages of three interleaved memory organisations: the S-access, the C-access and the C/S-access for pipelined vector processing. In the comparison, you should be concerned with the issues on effective memory bandwidth, storage schemes used, access conflict resolution and cost-effective tradeoffs.

Ans.

It is a technique for compensating the relatively slow speed of DRAM(Dynamic RAM). In this technique, the main memory is divided into memory banks which can be accessed individually without any dependency on the other.

For example: If we have 4 memory banks(4-way Interleaved memory), with each containing 256 bytes, then, the Block Oriented scheme(no interleaving), will assign virtual address 0 to 255 to the first bank, 256 to 511 to the second bank. But in Interleaved memory, virtual address 0 will be with the first bank, 1 with the second memory bank, 2 with the third bank and 3 with the fourt, and then 4 with the first memory bank again.

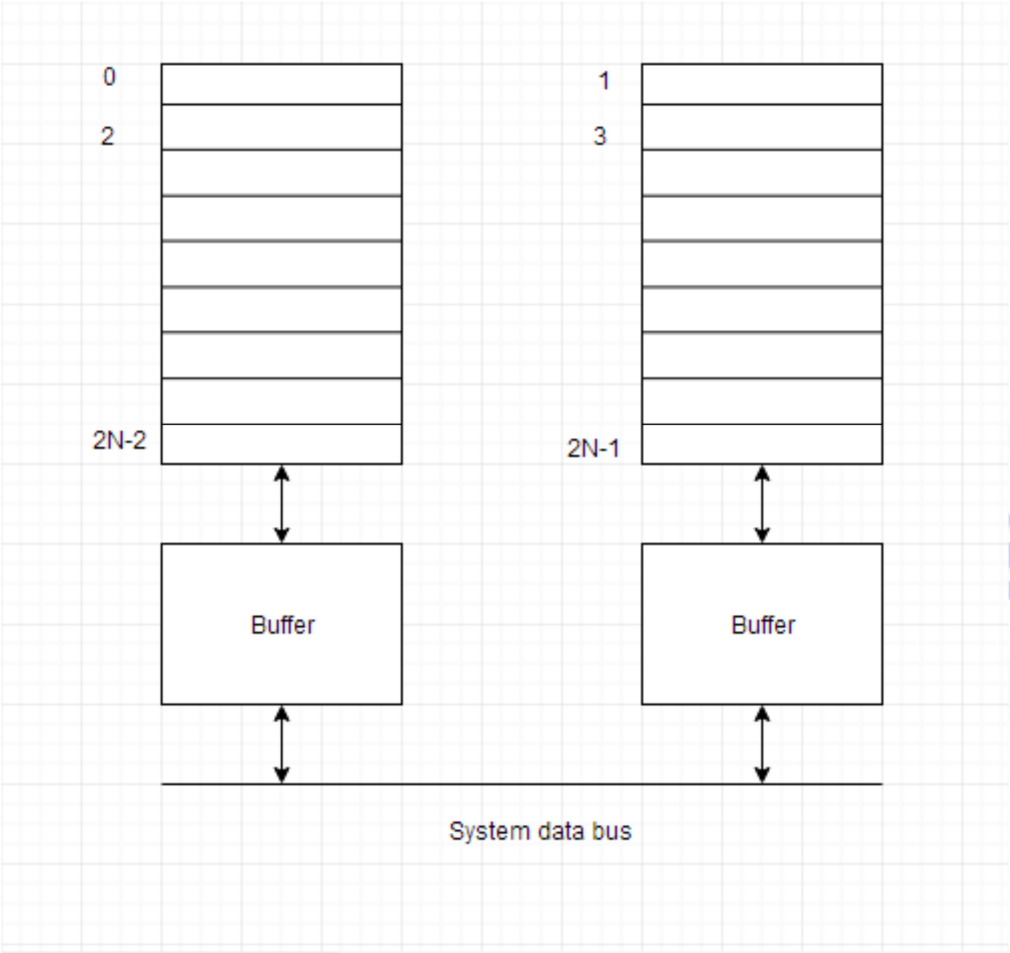
Hence, CPU can access alternate sections immediately without waiting for memory to be cached. There are multiple memory banks which take turns for supply of data.

Memory interleaving is a technique for increasing memory speed. It is a process that makes the system more efficient, fast and reliable.

An interleaved memory with n banks is said to be n-way interleaved. In an interleaved memory system, there are still two banks of DRAM but logically the system seems one bank of memory that is twice as large.

In the interleaved bank representation below with 2 memory banks, the first long word of bank 0 is floowed by that of bank 1, which is followed by the second long word of bank 0, which is followed by the second long word of bank 1 and so on.

The following figure shows the organization of two physical banks of n long words. All even long words of logical bank are located in physical bank 0 and all odd long words are located in physical bank 1.



Types of Interleaving

There are two methods for interleaving a memory:

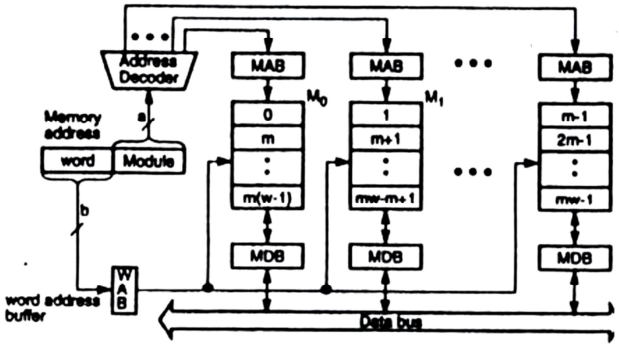
2-Way Interleaved

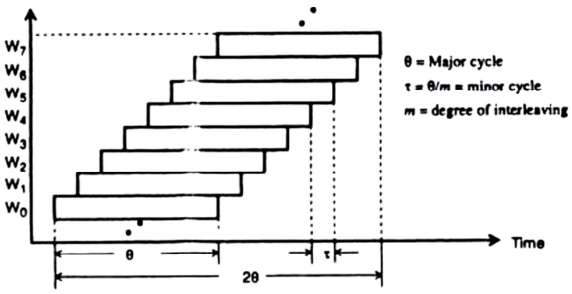
Two memory blocks are accessed at same time for writing and reading operations.

4-Way Interleaved

Four memory blocks are accessed at the same time.

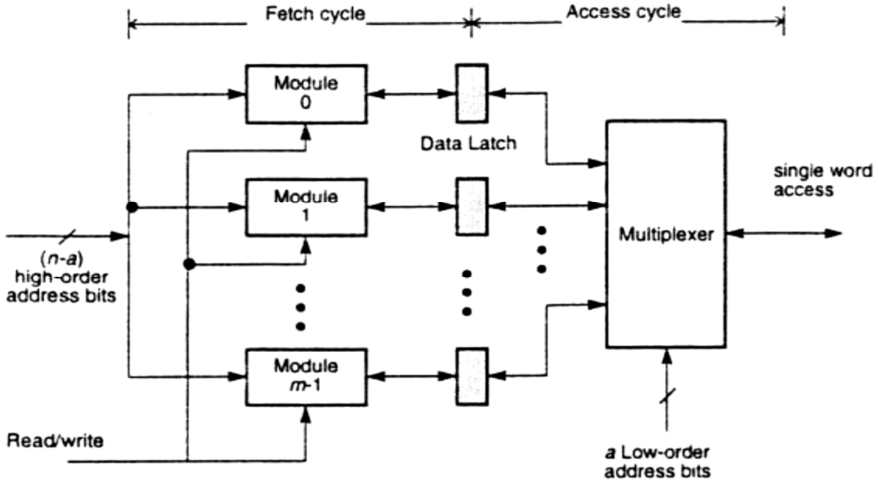
C Access Memory Organization:-





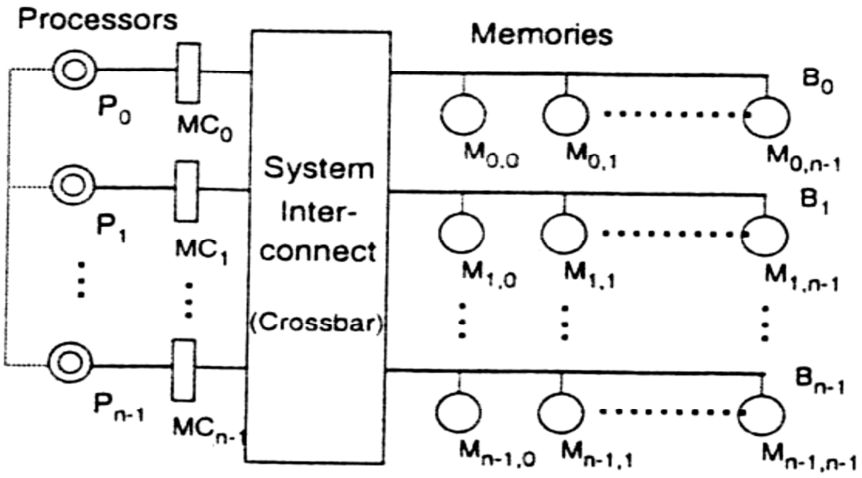
Vector access scheme from interleaved memory modules
• m-way low-order interleaved memory structure
• Allows m memory words to be accessed concurrently
• This is called C-access

S Access Memory Organization:-



Similar to low-order interleaved memory
– High order bits select modules
– Words from modules are latched at the same time
– Low order bits select words from data latches – This is done through the multiplexed with higher speeds (minor cycles)
• Allows simultaneous access • This is called S-access

C/S Acces Memory Organization:-



C-access and S-access are combined

• n access busses with m interleaved memory modules
• The m modules on each bus are m-way interleaved to allow C-access
• The n busses operate in parallel to allow S-access

Memory Bandwidth, Storage schemes used, access conflict resolution and cost effective tradeoffs.

:

Memory bandwidth has always been a critical factor for the performance of many data intensive applications. The increasing processor performance, and the advert of single chip multiprocessors have increased the memory bandwidth demands beyond what a single commodity memory device can provide. The immediate solution is to use more than one memory device, and interleave data across them so they can be used in parallel as if they were a single device of higher bandwidth. In this paper we showed that fine-grain memory interleaving on the evaluated many-core architectures with many DRAM channels was critical to achieve high memory bandwidth efficiency. Our results showed that performance can degrade up to 50% due to achievable bandwidths being far from the maximum installed.

On many commercial supercomputers, several vector register processors share a global highly interleaved memory in a MIMD mode. When all the processors are working on a single vector loop, a significant part of the potential memory throughput may be wasted due to the

asynchronism of the processors. In order to limit this loss of memory throughput, a SIMD synchronization mode for vector accesses to memory may be used. But an important part of the memory bandwidth may be wasted when accessing vectors with an even stride. In this paper, we present IPS, an interleaved parallel scheme, which ensures an equitable distribution of elements on a highly interleaved memory for a wide range a vector strides. We show how to organize access to memory, such that unscrambling of vectors from memory to the vector register processors requires a minimum number of passes through the interconnection network.

Using this approach, multiple conventional disks are grouped together and function as if they were a single one, with the data spread among the multiple disks and transferred in parallel. In this paper, we study four alternative implementations for achieving higher data rates in a disk subsystem, focusing on the trade-offs between the number of devices and the number of data paths, keeping the number of physical devices constant (which may keep the cost roughly constant). The performance advantages and limitations of the alternative implementations are analyzed using an analytic queuing model and compared to a conventional disk subsystem.

Question 2: Explain the following system features associated with the Illiac-IV, the BSP, and the MPP array processors: (a) Multi-array configurations of the Illiac-IV (b) The prime memory for the BSP (c) The bit-slice operations in the MPP (d) Concurrent scalar-array operations in the BSP (e) Concurrent I/O and arithmetic logic operations in the MPP (f) The staging memory configurations in the MPP (g) Host computers for the Illiac-IV, the BSP, and the MPP (h) The I/O facilities in the Illiac-IV, the BSP and the MPP

Ans.

The ILLIAC IV system was the first real attempt to contruct a large-scale parallel machine, and in its time it was the most powerful computing machine in the world. It was designed and constructed by academics and scientists from the University of Illinois and the Burroughs Corporation. A significant amount of software, including sophisticated compilers, was developed for ILLIAC IV, and many researchers were able to develop parallel application software.

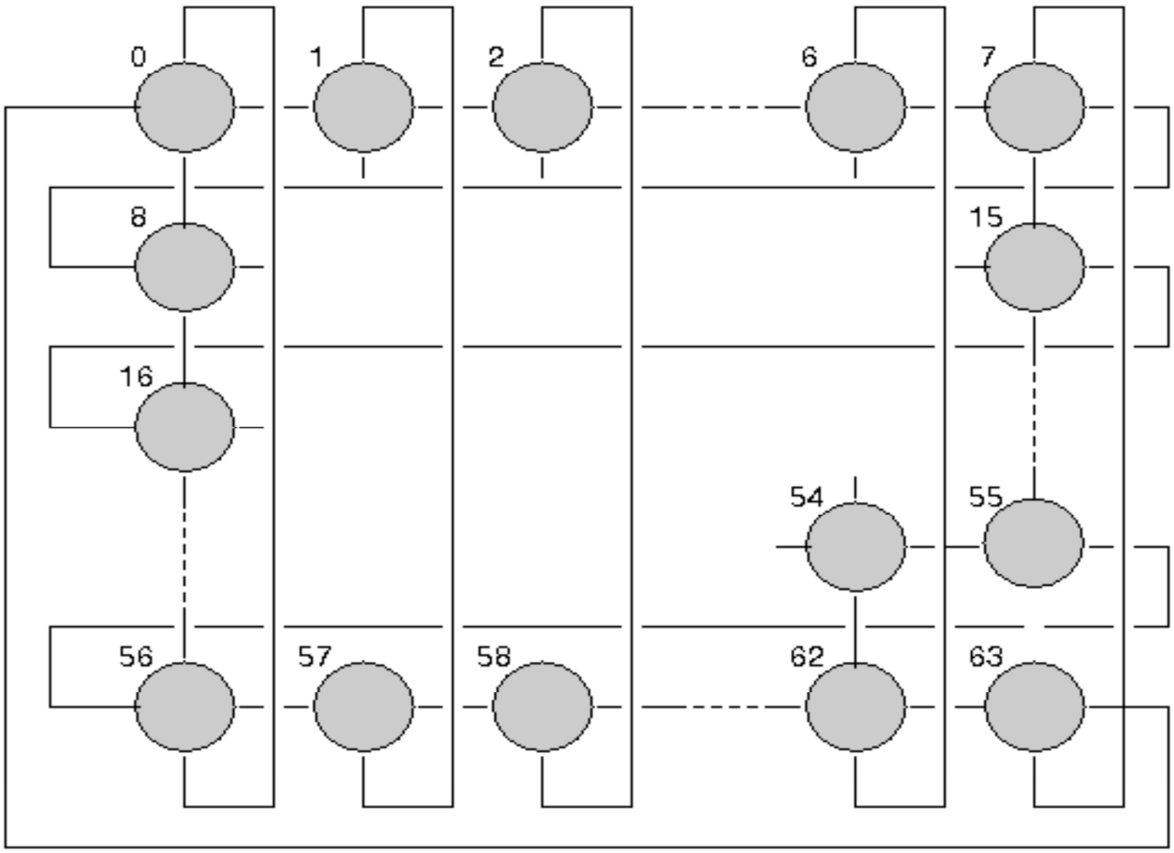
ILLIAC IV grew from a series of ILLIAC machines. Work on ILLIAC IV began in the 1960s, and the machine became operational in 1972. The original aim was to produce a 1 GFLOP machine using an SIMD array architecture comprising 256 processors partitioned into four quadrants, each controlled by an independent control unit. Unfortunately, as is often the case with such ambitious projects, escalating costs and unforeseen engineering problems resulted in just a single quadrant being built. The clock speed of the machine was intended to be 25 MHz but this too had to be reduced to 10 MHz, due partly to signal transmission delays resulting from the machine's large physical dimensions.

The processors in each quadrant were connected in the topology shown in the figure. Although this looks superficially rather like a square grid of connections it is in fact known as a *chordal ring* (see under Interconnection Networks/Static Networks), due to the shifted wrap-around of the boundary connections. Each inter-processor link consisted of a bi-directional 64-bit wide channel.

The control unit of ILLIAC IV was responsible for performing scalar operations and issuing SIMD instructions to an array of 64 processing elements. These elements executed instructions in lockstep, although each processing element had the ability to execute instructions conditionally using *local*condition variables. This mechanism whereby processing elements selectively "sit out" instructions makes the machine particularly flexible, and is a feature that has been included in all subsequent SIMD machines. It can even be seen in some vector machines in the form of control vectors.

Instructions for both the scalar section and the ILLIAC IV array were stored in the 2048 x 64-bit local memories associated with each processing element. These memories were constructed using thin-film storage devices and had access and cycle times of 120 and 240ns respectively. The control unit (CU) interface to these memories was a further example of array parallelism in operation; the data pathway between the CU and the memories was 512 bits wide permitting the CU to access one 64-bit word from each memory module in one *row* of processing elements concurrently (and at a common address), thus achieving an effective peak memory bandwidth of 1 word every 30 ns.

Although the actual performance of ILLIAC IV on real applications was only 2 to 4 times that of a CDC 7600, the machine is of significant historical value since it is arguably the origin of all subsequent parallel machines. Details of the architecture and of ILLIAC IV are given in Barnes *et al.*and an account of the development of the machine is presented by Falk .



BSP Host Computers, I/O operations, Concurrent scalar-array operations in the BSP

The Burroughs Scientific Processor (BSP) was effectively a successor to the ILLIAC IV machine, but with an architecture modified to reflect the fact that the BSP was intended to be a commercial product. It had fewer processing units than ILLIAC IV, just sixteen in the pre-production version, and most importantly these sixteen processors all enjoyed equal access to a common logical address space which was divided into a number of physically separate memory modules. The basic structure of the BSP is illustrated in the figure. Each processing element was nothing more than an arithmetic unit with input and output registers, and these units were homogeneous and non-pipelined.

The BSP was a 48-bit machine, and each arithmetic unit (AU) performed floating-point addition and multiplication in two 160 ns clock periods. The four units which constitute the array (the AUs, memories, result routing switch, and operand routing switch) formed a five-stage macro-pipeline, and by careful scheduling of micro-instructions the Array Control Unit (ACU) was able to overlap instructions in order to maximise the utilisation of the macro-pipeline. The BSP operated by partitioning multi-dimensional array operations between the AUs on an element-by-element basis. The ACU received instructions from the scalar processor and decomposed them into micro-operations which were then scheduled using templates. These were effectively pre-computed assignments of the five stages in the circular macro-pipeline to the micro-cycles within each instruction.

A typical sequence of micro-operations required to process each group of sixteen operands would be

read operands from memory

route operands to AUs

perform arithmetic operations

route results to their memory modules

write results to memory

The BSP provides equal access to all memory modules, from all arithmetic units, and as such is able to hide the array-like features of the machine from software. In practice the programming of the BSP, and the types of language structures most suitable for the form of parallelism it embodies, are reminiscent of vector processors. This is in fact true of all array processors with globally accessible operands, since parallel array units can be organised as simple cyclic arrays, and cyclic array structures have the same performance characteristics as pipeline structures. The BSP manages to achieve a high performance connection between an array of arithmetic units and an array of shared memory modules by a rather novel address interleaving mechanism, and this is worth considering in a little more detail.

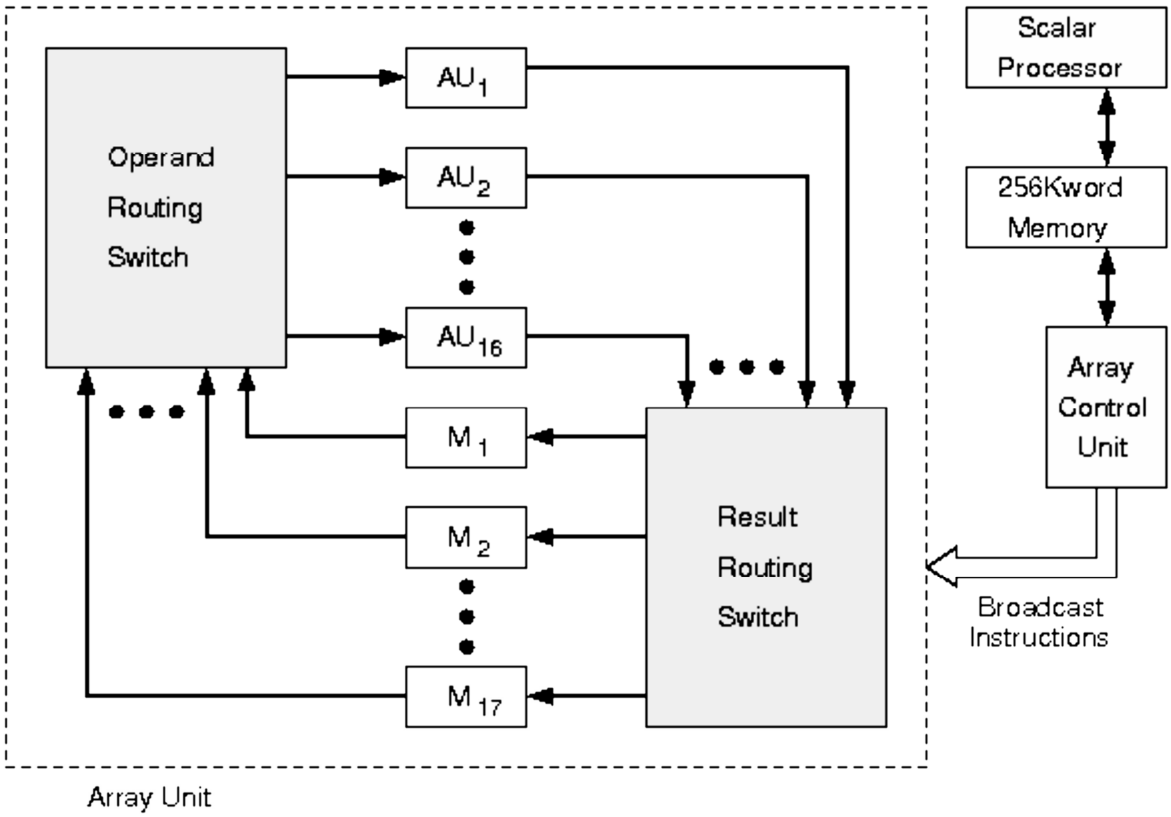
In the BSP the unit of parallelism is the vector, and elements of these vectors are accessed at index locations which can vary by a fixed increment. This increment may be any integer value, and this allows rows, columns and diagonals of multi-dimensional arrays which are mapped on to a BSP vector to be extracted by the ACU with ease. For example, a two-dimensional array X may be defined in Pascal notation as

X : array [1..column_length, 1..row_length] of real;

and in the BSP this would be laid out in memory in a column-wise manner. Therefore, to extract a column requires an inter-element stride equal to 1, and to extract a row requires a stride equal tocolumn length. Arbitrary diagonals can be extracted by using a stride equal to column_length + 1 or column_length - 1. High performance processing of these arrays is achieved by extracting sixteen elemental operand sets and presenting them to the sixteen arithmetic units in parallel, and naturally maximum throughput of the array can only occur if all elements are located in different memory modules. The interleaving scheme in the BSP therefore incorporated 17 memory modules, the lowest prime number greater than 16. For memory address a, the module number m containing that address is hence given by

$m = | a |17$

and the offset i within module m is given by



Bit Slicing operations in the MPP

Bit slicing is a technique for constructing a processor from modules of processors of smaller bit width, for the purpose of increasing the word length; in theory to make an arbitrary n-bit CPU. Each of these component modules processes one bit field or "slice" of an operand. The grouped processing components would then have the capability to process the chosen full word-length of a particular software design.

Bit slicing more or less died out due to the advent of the microprocessor. Recently it's been used in ALUs for quantum computers, and has been used as a software technique (e.g. in x86 CPUs, for cryptography.

Bit slice processors usually include an arithmetic logic unit (ALU) of 1, 2, 4, 8 or 16 bits and control lines (including carry or overflow signals that are internal to the processor in non-bitsliced CPU designs).

For example, two 4-bit ALU chips could be arranged side by side, with control lines between them, to form an 8-bit ALU (result need not be power of two, e.g. three 1-bit can make a 3-bit ALU,thus 3-bit (or n-bit) CPU, while such hasn't been used in volume). Four 4-bit ALU chips could be used to build a 16-bit ALU. It would take eight chips to build a 32-bit word ALU. The designer could add as many slices as required to manipulate increasingly longer word lengths.

A micro sequencer or control ROM would be used to execute logic to provide data and control signals to regulate function of the component ALUs.

Known bit-slice microprocessor modules:

1-bit slice

2-bit slice:

Intel 3000 family (1974), e.g. Intel 3002 with Intel 3001, second-sourced by Signetics and Inter sil

4-bit slice:

National GPC/P / IMP-4 (1973), second-sourced by Rockwell

National IMP-16 family (1973), e.g. IMP-00A/520D (RALU) with IMP16A/521D and IMP16A/522D, cascadable up to 16 bit

AMD Am2900 family (1975), e.g. AM2901, AM2903

Monolithic Memories 5700/6700 family (1974) e.g. MMI 5701 / MMI 6701, second-sourced by ITT Semiconductors

Texas Instruments SBP0400 (de) (1975), cascadable up to 16 bit

Texas Instruments SN74181 (1970)

Texas Instruments SN74S281 with SN74S282

Texas Instruments SN74S481 with SN74S482 (1976)

Fairchild 9400 (MACROLOGIC), 4700

Motorola M10800 family (1979), e.g. MC10800

8-bit slice:

National IMP-8 family (1974), cascadable up to 32 bit

Texas Instruments SN54AS888 / SN74AS888

Fairchild 100K

ZMD U830C (1978/1981), cascadable up to 32 bit

16-bit slice:

AMD Am29100 family

Synopsys 49C402

Primary memory in BSP

A board support package (BSP) is essential code for a given computer hardware device that will make that device work with the computer's OS (operating system). The BSP contains a small program called a boot loader or boot manager that places the OS and device drivers into memory. The contents of the BSP depend on the particular hardware and OS.

Specific tasks that the BSP performs include the following, in order:

- Initialize the processor.
- Initialize the bus.
- Initialize the interrupt controller.
- Initialize the clock.
- Initialize the RAM (random access memory) settings.
- Configure the segments (if applicable).
- Run the boot loader.

In addition to the foregoing, a BSP can contain directives, compilation parameters, and hardware parameters for configuring the OS.

MPP

MPP (massively parallel processing) is the coordinated processing of a program by multiple processor s that work on different parts of the program, with each processor using its own operating system and memory . Typically, MPP processors communicate using some messaging interface. In some implementations, up to 200 or more processors can work on the same application. An "interconnect" arrangement of data paths allows messages to be sent between processors. Typically, the setup for MPP is more complicated, requiring thought about how to partition a common database among processors and how to assign work among the processors. An MPP system is also known as a "loosely coupled" or "shared nothing" system.

An MPP system is considered better than a symmetrically parallel system (SMP) for applications that allow a number of databases to be searched in parallel. These include decision support system and data warehouse applications.

www.ignousite.blogspot.com
Sunil Poonia
Estd : 2014
IGNOU STUDY HELPER