# INTERCONNECTION NETWORK ANALYSIS AND LOGIC DESIGN

INFORMATICA 2/89

Keywords: parallel processing, interconnection networks, asynchronous vs. synchronous timing, logic design

Andrej Žagar, Peter Brajak
Iskra Delta, Ljubljana

This paper shows the development efforts in designing a interconnection network for 64 processor tightly coupled PARSYS parallel machine. In the first two chapters a network analysis is described based on the discrete statistical simulation model. From the results of the discrete statistical simulation model a high performance asynchronous Routing Node logic design is defined and implemented for the n-cube based interconnection network.

## 1. INTRODUCTION

We are entering the era of fundamental changes in the field of computer architecture. In particular, a large part of these changes have been represented by a transition from serial to parallel processing. This transition is stimulated by at least four reasons:

- the cost of digital hardware has dropped to the point that processors need not be considered the scared resources,
- improvement of performance of monoprocessors is technologically limited,
- the scope of problems whose algorithmic complexity exceeds the capability of todays most powerful computers is very large,
- the formulation of the problems naturally suggest parallel realization.

Market Research Group, a consulting company that follows the trends on computer market, foretells that by the year 1990, 48% of all large scale computers will have one of the forms of parallel processing. Most of the developed countries support parallel system projects (Japan, German Federative Republic, USA, etc.). These projects differ from technological to philosophical concepts. However, one is common to all of them: the proposed computers are order of magnitude faster than conventional computers, they are successful, and there is a fully new open market for the application software.

### 1.1. RATIONALE AND CONCEPTS

PARSYS[12,13] is a tightly-coupled MIMD (multiple instruction, multiple data) research and development project on parallel processing at Iskra Delta Computers. Specifically, the project is involved with:

- development of the first prototype hardware and system software,
- development of specific program development environments,
- development of application software.

The rationale for the decision to build a MIMD tightly-coupled parallel system rather than a super fast SIMD (single instruction, multiple data) vector machine is multifold:

- the market for MIMD multiprocessors is just beginning to evolve, whereas for vector machines it is highly competitive,
- the higher flexibility of MIMD machines allows for a broader application spectrum,
- the development cost of the MIMD based multiprocessor system is lower than of the vector machine, since it is possible to use of-the-shelf standard components,
- the whole spectrum of new software products could benefit from general purpose MIMD machines (vector processors are too much specialized).

The PARSYS project is product-oriented. This means that in contrast to pure research and prototype development, an additional market-oriented requirements must be satisfied such as:

- the desired absolute performance must be obtained at the competitive cost-effectiveness,
- the machine must be manufacturable, testable and maintainable,
- there must exist a time schedule during which the research and development must lead to a production model.

In order to meet the time schedule, the architectural design of the PARSYS machine has been based not only on the innovative concepts but also on the solutions that have incorporated already proven technologies, methodologies, standards and products.

The recent research and development work on the PARSYS system has been focused on:

1. top level architectural specification and implementation a 64 processor machine based on multistage interconnection topology,
2. implementation of minimal kernel for fast process communication,
3. development of software tools for automatic parallelization and debugging.

In this paper we show only the hardware realization of the PARSYS n-cube interconnection network. The work was done at Iskra Delta Computers as partial fulfillment for Bs.EE. degree[11] from June 1987 to September 1988.

## 2. INTERCONNECTION NETWORK ANALYSIS

Interconnection networks for concurrent computers have been studied intensively, and many different topologies have been proposed: from non-blocking, but impractical crossbar switches, Benes and Batcher networks to more practical blocking networks such as omega, flip, delta, indirect binary n-cube.

PARSYS original network topology used flip multistage interconnection network[1], directly transformed into the binary n-cube. The benefit of the transformation was the simplicity of the n-cube implementation and the versatility of the multistage interconnection network.

PARSYS machine can be functionally represented as in figure 2.1. Programs and data can be either stored in processor's local (LM) or global memory (GM). CPU-s can access global memory only through the interconnection network. The problem is that the time delay required to access global memory is much larger than the time to access local memory. In order to decrease traffic in the network and to increase overall program execution it is required that both program code segments and parts of data segments (local stack) are placed in the local memory.

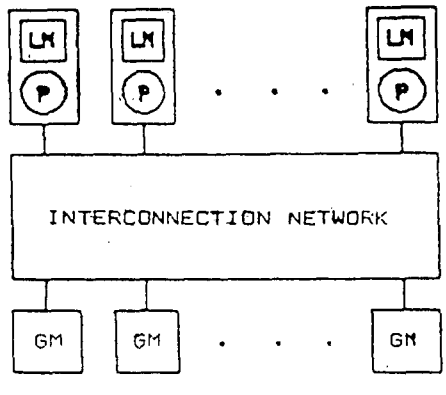


Figure 2.1. The parallel computer with distributed global and local memory

The interconnection network analysis must take advantage of the fact that the traffic flow is greatly reduced by introducing the local memories in the system and that a simple network design can sufficiently absorb traffic flow and bring global memory request in the fastest possible way.

### 2.1. THE PURPOSE OF INTERCONNECTION NETWORK

The interconnection network statistical analysis and the interconnection network simulation introduced in this paper are valid for any multiprocessor interconnection network topology which is based on global/local architecture. The goal of these two chapter is to show transport characteristics in the network and to make a ground for the optimal multiprocessor network logic design based on the results of the analysis .

CPU access global memory by creating a read or write packet. Packets travel through the network using self-routing mechanism described in [1].

The read packet contains an address and a routing tag when it travels from the processor to the destination global memory module (route: P -> GM) and it consists of data and the same routing tag when it returns back to the same processor (route: GM -> P).

The write packet consists of an memory address, a routing tag and data. It travels only in one direction: from the processor to the destination global memory module (route: P -> GM).

### 2.2. INTERCONNECTION NETWORK STATISTICAL ANALYSIS

It is very hard to determine statistically the percentage of global read and global write memory accesses. However, an average and a maximum number of packets in the network are very valuable information for the optimization process of the overall network design.

When statistical features and relative time delays have been included in the model, the basic network's features can be restored.

Fortunately, we can use some statistics gathered from the uniprocessor systems and tune them for the analysis of the parallel interconnection design. The results of statistical analysis of computer programs for IBM mainframe machines[6] have shown the following characteristics:

1. a typical program may be represented by the finite automata,
2. instructions may be divided in three characteristic groups, each group representing a node in the automata,

- in the first group are instructions like INA, ASLA and CLRA. These instructions are register only instructions do not require memory access, PC := PC+1,

- memory jump instructions (JMP, JSR,..), PC := <new address>

- read and write instructions (LDA, STA,..), Reg := Memory[Address]; PC := PC+1;

3. there exists a state transition among the groups: (figure 2.2.1)

- probability, that an instruction from the first group is followed by an instruction in the same group is 11/20,

- probability that an instruction from the first group is followed by an instruction from the second group is 1/4,

- probability that an instruction from the first group is followed by an instruction from the third group is 1/5.

- the similar relations are for the second and for the third group, as well.

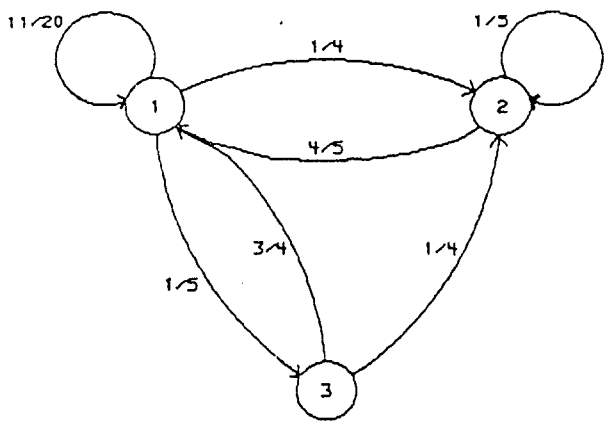


Figure 2.2.1. Probability scheme of the instruction cycle

4. when the memory space is divided to the local and the global address space, then only every tenth instruction will require global memory request for its execution.

Since we are interested only at the memory requests that require global memory, we can compress state 1 and state 3 into only one state, called local state, and state 2 called global state.

A probability that machine will turn from the local state to the global state is 1/10.

The following assumptions are necessary to construct a statistical model of the multiprocessor network based on previously defined probabilities:

- programs are all loaded into local memories,

- there is an equal number of read and write instructions,

- a probability scheme of the CPU machine cycle is equal to the probability scheme of the instruction cycle,

- the CPU's machine cycle is equal to the processor's machine cycle,

- the processor which has just sent a read packet to the network has to wait until this packet returns,

- the processor has finished a write cycle when the write packet had been sent in the network,

- PreRouting and PostRouting delays are treated as a part of the network delay,

- the processors are treated as a two state machines,

- the processors are statistically independent,

- network is deadlock free,

- there is no combining in the network necessary[7].

From the statistical analysis and the assumptions previously stated, it can be concluded that the processor in 20 machine cycles generates one read and one write packet.

The following symbols should be defined:

- N is a number of processors,
- Tcpu is the CPU'S machine cycle,
- Tn is an average packet delay in the network,
- Tm is a global memory module's access time,
- Tw is time which is needed to write data in the destination global memory module,
- Tr is time which is required to read data from the global memory module and
- Np is a number of packets in the network.

Therefore:

$$Tr = Tn + Tm + Tn. \qquad 2.2.1.$$

$$Tw = Tn + Tm. \qquad 2.2.2.$$

## 2.3. INTERCONNECTION NETWORK SIMULATION

A discrete simulation model (figure 2.3.1.) is constructed for the three functional parts:

- (CPU(Td)) delay units represent the processors,
- (WPGM(Ta), RPGM(Tb) and RPCPU(Tc)) delay units represent the packets in the multiprocessor network and
- (GM(Te)) delay units represent the global memory.

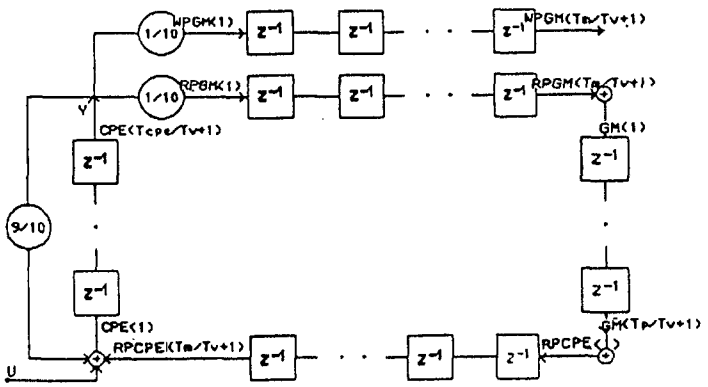


Figure 2.3.1.

The number of packets which are sent to the network at the moment T, is depended on the number of processors which finished their last machine cycle in the moment (T-Tcpu). The read packets (RPGM(Tb) delay units) and the write packets (WPGM(Ta) delay units) travel through the network to the global memory modules (route: P -> GM). When the write packets come to the global memory modules, write data in the memory and their route is finished there. On the other side, the read packets read data from the global memory modules (GM(Te) delay units) and after that, they return to the processors with required data (RPCPU(Tc) delay units, route: GM -> P).
Exact definitions are:

N * WPGM(Ta) - a number of write packets which are Ta time units in the network,

N * RPGM(Tb) - a number of read packets which are Tb time units in the network (route: P -> GM),

N * RPCPU(Tc) - a number of read packets which are Tc time units in the network (route: GM -> P),

N * CPU(Td) - a number of processors which are going to start new machine cycle for (Tcpu/Ts + 1 - Td) time units,

N * GM(Te) - a number of global memory modules which are going to return read packets in the network for (Tm/Ts + 1 - Te) time units and

Ts - is a sampling time.

The number of all packets in the network (Np) is a sum of the read and the write packets which are in the network at the particular moment:

$$Np = N * \sum_{i=1}^{Tn/Ts+1} (WPGM(i) + RPGM(i) + RPCPU(i)). \qquad 2.3.1.$$

The initial condition is a delta impulse on one of the processors' delay units CPU(Ti). The simulation will start after (Tcpu/Ts + 1 - Ti) time units.

The probability scheme is valid for the first machine cycle too.

All processors could send packets to the network simultaneously. However, this possibility is one of the initial conditions too. If this condition could happen, the network would congest mostly. The maximum number of packets in the network is:

- N, if 0 < Tn < Tcpu and

- k * N, if (k - 1) * Tcpu < Tn < k * Tcpu. 2.3.2.

System function is:

$$\frac{Y(Z)}{U(Z)} = \frac{Z^{-Tcpu/Ts}}{1 - 19/20 * Z^{-Tcpu/Ts} - 1/20 * Z^{-(Tcpu+Tn+Tm+Tn)/Ts}} \qquad 2.3.3.$$

The sampling time is bounded by the upper and the lower limit. The upper limit is determined by:

$$Tcpu = a * Ts,\quad Tn = b * Ts \text{ in } Tm = c * Ts, \qquad 2.3.4.$$

where a, b and c are integer. If the condition is not accomplish, the extended Z-transformation must be used and the model can not be simulated on digital computers.

The lower limit is defined by the number of the delay units which represent the processors. The maximum number of CPU(Ti) delay units is N. This is the highest asynchronization which can be hardly achieved with N processors.

The sampling time is determined by the requested precision and the observing range.

y(k) is defined by the inversive Z-transformation:

$$y(k) = \mathcal{Z}^{-1}( Y(Z) ). \qquad 2.3.5.$$

The number of all packets in the network (Np) is:

$$Np(k) = N * ( 1/20 * \sum_{i=k-b}^{k} y(i) + 1/20 * \sum_{i=k-b}^{k} y(i) + 1/20 * \sum_{i=k-b-c-b}^{k-b-c} y(i) ) \qquad 2.3.6.$$

for T = k * Ts.

The analytic analysis is very complicated so the discrete simulation is used.

The number of all packets in the network (Np) after the transitional phenomenon is found out by the simulation algorithm given in appendix I.

The simulation results are shown in the figure 2.3.2..

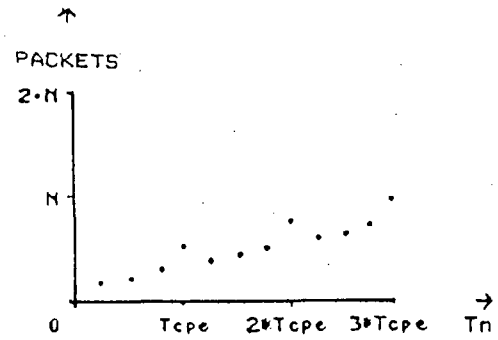


Figure 2.3.2. The number of all packets in the network (Np) after the transitional phenomenon (Tcpu = 125 ns, Tn = 25 - 300 ns, Tm = 100 ns, Ts = 5 ns)

The number of packets increases with Tn. However, the non-regularity shows up at the condition:

$$Tn = k * Tcpu \; ; \; k = 1,2,.. \qquad 2.3.7.$$

This non-regularity must be bypassed. This can be assured by enforcing asynchronous processor operations.

The maximum number of all packets in the network (Npm) is found out by the simulation algorithm given in appendix II.

The simulation results are shown in the figure 2.3.3..

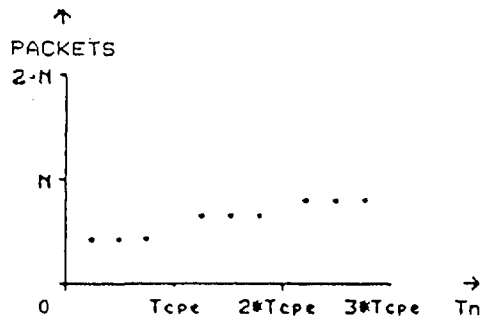


Figure 2.3.3. The maximum number of all packets in the network (Tcpu = 125 ns, Tn = 25 - 300 ns, Tm = 100 ns, Ts = 5 ns)

A comparison between the equation 2.3.2. and the figure 2.3.3. shows that the maximum number of all packets in the network depends on the initial condition of the simulation. Furthermore, the maximum number of packets is in the network when processors work synchroniously.

Therefore, the simulations show that hardware asynchronization of the whole system (processors among each other and network itself) must be assured.

An efficient network can be built if only Tm is slightly lesser than k * Tcpu.

The results of the interconnection network statistical analysis based on the discrete simulation model get together with the results of other analysis which is built on a interconnection network as an arbitration system. The advantage of the described analysis is also that it is independent from the system architecture.

The main disadvantages of this analysis is that Tm must be presumed to find out the exact number of packets in the network. However, the exact value of Tm parameter has no essential influence on the network optimization and logic design.

## 2.4. MULTISTAGE N-CUBE OPTIMIZATION

Multistage networks like OMEGA, DELTA, BUTTERFLY, FLIP and BINARY N-CUBE are well known interconnection networks[14]. The main advantage of these networks is that a processor can access every memory module in $Log_N$ steps. The main disadvantages of these networks is the number of crossing links which increase with every new level and the processor inability to access local memories in less than $Log_N$ steps.

The FLIP network (figure 2.4.1.) has an isomorphic characteristic which enables a transformation from FLIP's multistage structure into $Log_N$ dimensional n-cube structure[1].

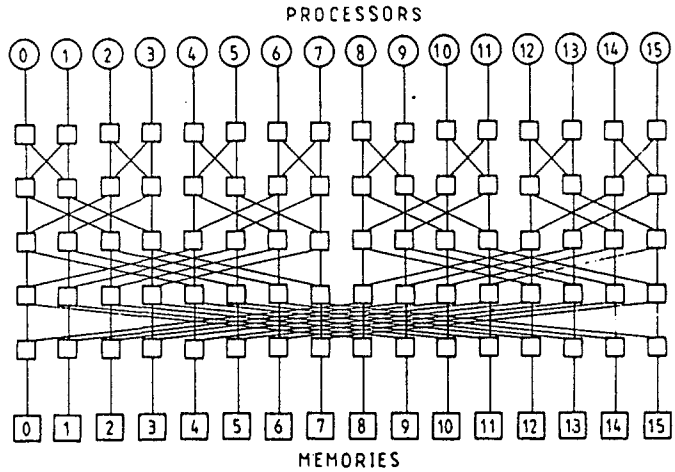


Figure 2.4.1.

A n-cube vertex consists of a processor, the arbiters under that processor and a global memory module under that arbiters.

A packet, which arrives to the arbiter, is stored in its memory block. After that a packet's route is determined by its routing tag. If the route is ready for the transfer, the packet is copied in the neighboring arbiter, processor or global memory module. Only one packet can be carried from the source (arbiter's memory block) to the sink (neighboring memory block) in the one arbitration cycle. The arbitration cycle's length is named an arbitration time (Ta) which is proportional with the average packet delay in the network (Tn). The relation between these characteristic parameters follows from a complex statistical analysis of arbitration systems. But only the fact that the arbitration time has to be as short as possible is important for the network logic design.

Some calculations are given in this chapter to be used as the direction for the final optimization. All calculations are made for 64 multiprocessor system.

A physical and logical ampleness, which is an optimization subject, is mostly depended on:

- one or bidirectional packets buses,
- packet buses width,
- the number of the arbiters and memory blocks,
- the kind of memory blocks.

Every vertex is connected with $Log_2N$ neighboring vertices. The number of all one-directional packet buses is:

$$N * Log_2N = 384 \text{ (buses)} \qquad 2.4.1.$$

The bidirectional bus needs the arbiter to apportion the bus to one of the sources. The number of all arbiters can increase by:

$$N * Log_2N / 2 = 192 \text{ (arbiters)} \qquad 2.4.2.$$

The one-directional bus requires more links in the bus, but the logic ampleness will decrease because the additional arbiters are not required.

Therefore, the read packet should include:

- 32-bit address or data,
- routing tag ($Log_2N$ = 6 bits),
- direction indicator (1 bit) and
- read or write mark (1 bit).

The write packet consists of:

- 32-bit address,
- data (32 bits),
- routing tag ($Log_2N$ = 6 bits),
- direction indicator (1 bit) and
- read or write mark (1 bit).

The maximum packet's width is 72 bits and it has to be equal the packet buses width. But the buses width is halved to achive a propitious rate between the network efficiency, logic ampleness and number of links. Therefore, the write packet is divided into two words:

The first word contains:

- routing tag,
- direction indicator and
- read or write mark,
- address.

After that, the second word is sent to the memory block and it consists only:

- data part.

The number of arbiters in the vertex is very high, so the reduction is required, too. The next architectures has been analyzed:

- $Log_2N$ stage n-cube (figure 2.4.2.),
- $Log_2N$ / 3 stage n-cube (figure 2.4.3.) and
- singlestage n-cube (figure 2.4.4.).

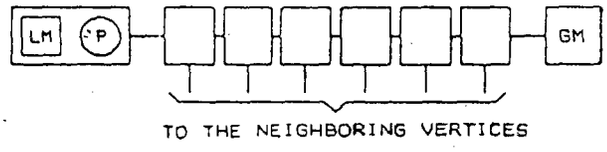


Figure 2.4.2.

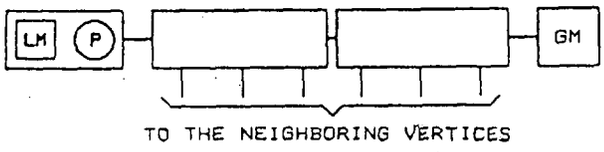


Figure 2.4.3.

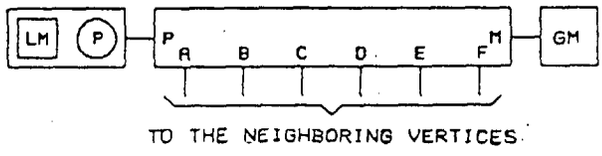


Figure 2.4.4.

The number of arbiters is:

- $N * Log_2N$ (arbiters)

  for $Log_2N$ stage n-cube,

- $N * Log_2N / 3$ (arbiters)

  for $Log_2N$ /3 stage n-cube and

- N * 1 (arbiters)

  for singlestage n-cube.

2.4.3.

The number of memory blocks is equal to the number of all arbiter sources or sinks and it is:

- $3 * Log_2N * N = 1152$ (memory blocks)

  for $Log_2N$ stage n-cube,

- $5 * Log_2N / 3 * N = 640$ (memory blocks)

  for $Log_2N$ /3 stage n-cube and

- $8 * N = 512$ (memory blocks)

  for singlestage n-cube.

2.4.4.

A following estimation could be given between the average packet delay, the arbitration time and the architecture:

- $Tn = Log_2N * Ta$

  for $Log_2N$ stage n-cube and

- $Tn = Ta * Q$

  for singlestage n-cube.

where is $Q > 1$. 2.4.5.

The estimation is based on the multistage networks without dead locks and races. If almost all interested data for processor i are placed in the memory module i, the Q is kept low.

The number of memory blocks is much higher than the average or maximum number of packets in the network (figure 2.3.2., figure 2.3.3. and equation 2.4.4.). Also the physical and logical ampleness decreases with the architection reduction (equation 2.4.3. and equation 2.4.4.). The third indicator is equation 2.4.5. which shows that Ta could be higher in the singlestage n-cube.

It is evident that singlestage n-cube is the most appropriate architecture and its efficiency is depended on:

- data disposition in the global memory modules,
- packets' conflict in the network.

Programs' adaptations to the architecture could mostly sooth this problems, so the single-stage n-cube is used for the network logic design.

Registers are used as memory blocks, because the number of all memory blocks is much higher then the maximum or average number of packets (figure 2.3.2., figure 2.3.3. and equation 2.4.4.). The register is formed of two blocks, because it has to store all packet and not only one word. If other way had been used, the logic ampleness would enormously increase. A solution is given in the figure 2.4.5..

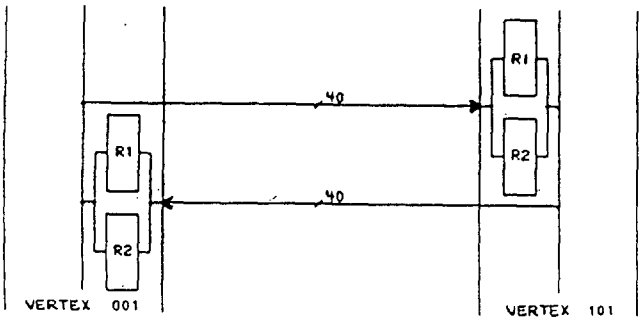


Figure 2.4.5.

At the beginning of the arbitration cycle the first word is carried in the block R1. After then the words are changed on the packet bus and after all the second word is transferred in the block R2.

Let's resume the optimization process and define the routing node' features:

- the network burden is lower if the asynchronous mode is used (equation 2.3.2. and figure 2.3.3.),
- Tn is proportional with Ta (follows from the arbitration theory)
- one-directional buses should be used,
- the buses width is 40 bits, therefore the write packets are divided into two words,
- singlestage n-cube is applied and
- registers are used as memory blocks.

# 3. ROUTING NODE LOGIC DESIGN

The routing node contains one arbiter which has eight sources and sinks. It is formed of:

- eight memory blocks,
- a synchronization logic,
- an arbitration logic and
- a termination logic.

A packet which comes to the routing node is stored in the register. After then the packet route is determined by the routing tag. If the next memory block on the packet route is empty, the bus request is generated. All bus requests from the sources are indicated in the synchronization logic. The arbitration logic chooses one of the bus requests and enables the packet transfer. The termination logic ends the arbitration cycle.

The routing node is built by 74F and 74AS logic families which are often used in application to achive high system performance. But on the other hand they are easy for designing because many equipment and software tools have been developed for these integrated circuits.

## 3.1. MEMORY MODULE

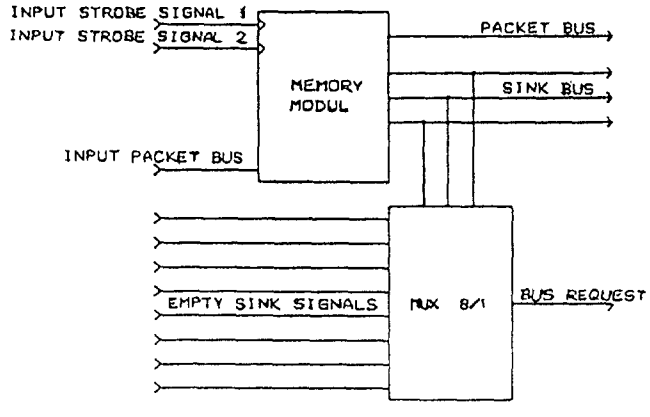


Figure 3.1.1.

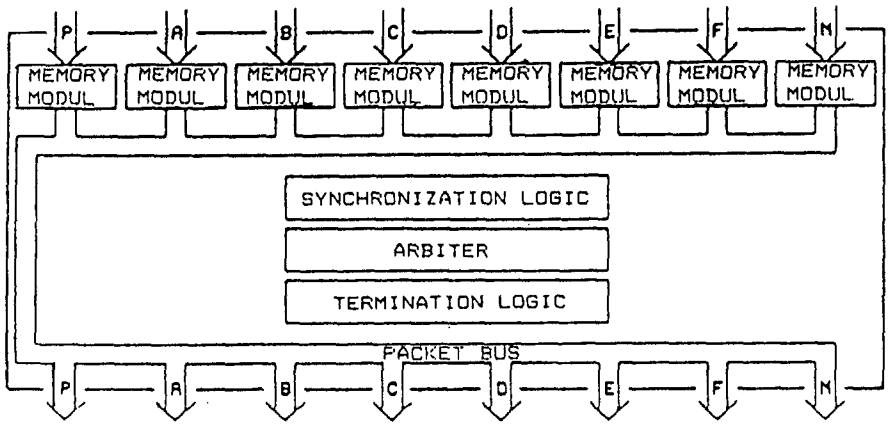


Figure 3.1. Routing node

The functions of the memory block are:

- to store packet in the registers,
- to determine the packet's length and directions,
- to define the sink,
- to generate a bus request.

A packet is written in the register by the input strobe signals. The sink is defined by the routing tag and a code of that memory block (P, A, B,..).

The arbiter on the level i is shown in the figure 3.1.2..

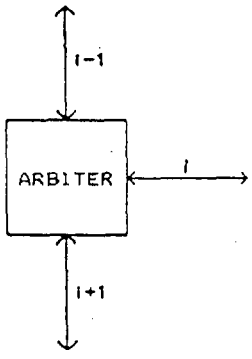


Figure 3.1.2.

When the packets travel from processors to memory blocks, the source can be the arbiter on the level i-1 or the neighboring arbiter on the level i.

For the source i-1 the sink is defined by the $r_i$ bit ($r_5$ $r_4$ $r_3$ $r_2$ $r_1$ $r_0$ is the routing tag):

$r_i$ = 0; the sink is the arbiter on the level i+1 and

$r_i$ = 1; the sink is the arbiter on the level i.

For the source i packets are transferred to the level i+1.

When the read packets return to processors, they reach the arbiter on the level i through the source i-1 and i.

For the source i+1 the sink is determined by the $r_i$ bit:

$r_i$ = 0; the sink is the arbiter on the level i-1 and

$r_i$ = 1; the sink is the arbiter on the level i.

For the source i packets are carried to the level i-1.

The routing tag is defined in the processor and it is:

routing tag =

= processor num. XOR global memory module num.

But the singlestage n-cube requires an extensive analysis of the routing tag.

Let's look the following example for the $Log_N$ stage n-cube:

A packet from the processor eleven travels to the global memory module three. The routing tag is :

1011 XOR 0011 = 1000.

A packet route is shown on the figure 3.1.3..

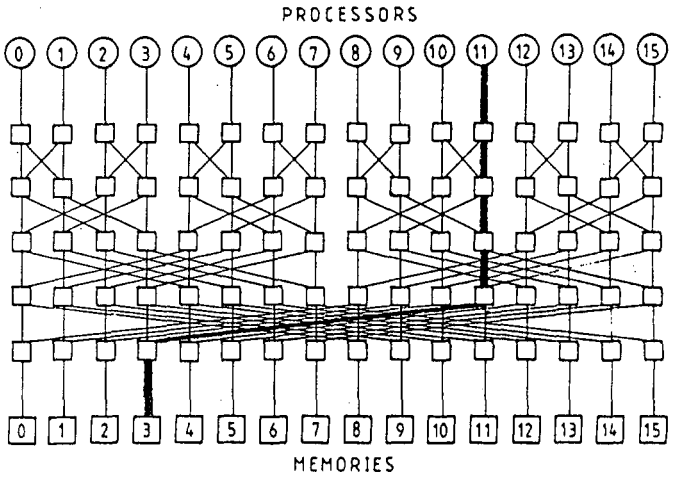


Figure 3.1.3.

The packet travels straight down through the arbiters, and it turns on the last level arbiter. However, in the singlestage n-cube, with only one arbiter between a processor and memory module, the packet is directly transferred to the last level - on the sink D, where it is received by the neighboring routing node (Figure 2.4.4. for 16 P - there is no sinks E in F).

Therefore, for the source P, the pattern of successive zeros from the $r_0$ inclusively in the routing tag has meaning for further route inside the node ( the route P -> GM). The source A can send packets in the both directions. For the route P -> GM is important the pattern of successive zeros ahead the $r_0$ and for the route GM -> P all packets are carried to the sink P. For the source B is important the pattern of successive zeros ahead the $r_1$ for the route P -> GM and the content of the $r_0$ for the route GM -> P. Let's explain the routing tag in the source D for the complete notion. The source can also send the packet in two directions. For the direction P -> GM is important the pattern of successive zeros ahead $r_3$ and the same pattern from $r_2$ backwards (the route GM -> P).

The function:

SINK = f(SOURCE, DIRECTION, ROUTING TAG) 3.1.1.

is given in the [11].

Functions like that are usually calculated by the state machine and the algorithm for it is shown in the figure 3.1.4..

Because state machines are very slow, the function is calculated by the decoder. The decoder has also to code the sink and to define the packet length. The sink code is separated from the sink bus by three-state gates. The sink has to be calculated very fast, because the bus request needs this information for its activation. The bus request is generated by the MUX. The MUX is addressed by the sink code and its inputs are signals which define if the sink is empty or not.

```
begin
     i:=f(level);
     case direction of
          "F":begin
                sink:=source+1;
                a:=0;
                while r(i+a)=0 do
                     begin
                          sink:=sink+1;
                          a:=a+1
                     end
              end;
          "B":begin
                sink:=source-1;
                a:=1;
                while r(i-a)=0 do
                     begin
                          sink:=sink-1;
                          a:=a+1
                     end
              end
     end
end.
```

Figure 3.1.4.

To achive high activation speed the bus request is generated on two different ways:

M1 = input strobe signal *

* f(SOURCE, DIRECTION, ROUTING TAG) *

* empty sink and

M2 = full memory block *

* f(SOURCE, DIRECTION, ROUTING TAG) *

* empty sink.

Therefore:

bus request = M1 V M1.

M1 is active when the packet is being writing in the memory block while M2 conforms and keeps the bus request active.

## 3.2. SYNCHRONIZATION LOGIC

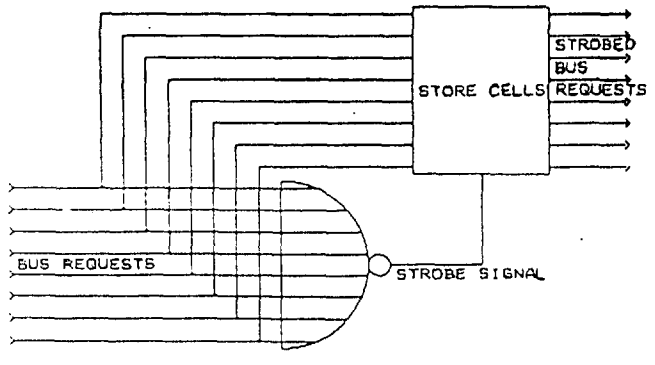


Figure 3.2.1. Synchronization logic

The synchronization logic is used to assure one decision in one arbitration cycle. To achive this purpose the logic has locked after the first bus request had come.

New request may come between locking but it is not important if they go through the logic or not. All bus requests which come later are waiting for the next arbitration cycle. If the lock does not exist, more than one decision can be possible in one arbitration period. This is undesirable in any case.

There are two possibilities how to make the lock:

- synchronous lock (the sampling is periodical) and
- asynchronous lock (the sampling is caused by the becaming signals).

The asynchronous lock is much faster and its application in the synchronization logic is shown on the figure 3.2.2.).

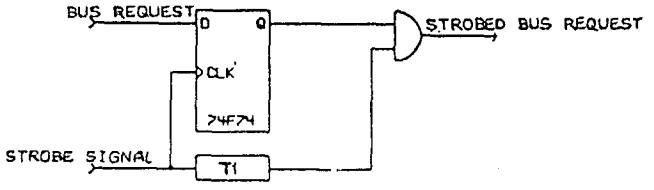


Figure 3.2.2.

A sampling signal is a disjunction of all bus requests. The next sampling is prohibited until only one bus request helds up in the active state. Because of that all bus requests can only be deleted by the termination logic.

The OR-gate has also a task to make a delay between the sampling signal and the bus requests, so this way the T-set is assured.

All sampling bus requests are conformed by the control signal, which is delayed for T1 seconds. The control signal delay has to prevent only the oscillations of sampling bus requests. Because packets which are late can wait for the next decision cycle or can be considered in this arbitration. Because of this reason the delay is only a bit longer than the maximum propagation delay through the cell.

## 3.3. ARBITRATION LOGIC

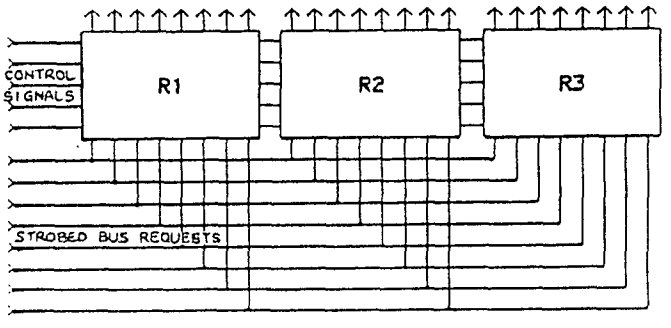


Figure 3.3.1.

An arbitration logic chooses one of the sampling bus requests and allows the packet transfer. A full source which has been processed in the last arbitration cycle has no priority ahead the other full sources. Information about the last processed source is stored until the next arbitration cycle. If more then one bus requests have occurred, the nearest right neighbor of the last processed source in a chain will be chosen. The chain is shown in the figure 3.3.2..

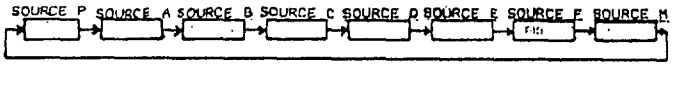


Figure 3.3.2.

If the source B is empty, the source C follows the source A.

The fast arbitration is achieved by three parallel decision logic. This is necessary because the logics are activated in different time.

They are:

- R1 enables the first word transfer with opening the three-state gates on the packet bus,
- R2 enables the second word transfer with opening the three-state gates on the packet bus and
- R3 is active across all arbitration cycle and it enables the sink code to occupate the sink bus.

## 3.4. TERMINATION LOGIC

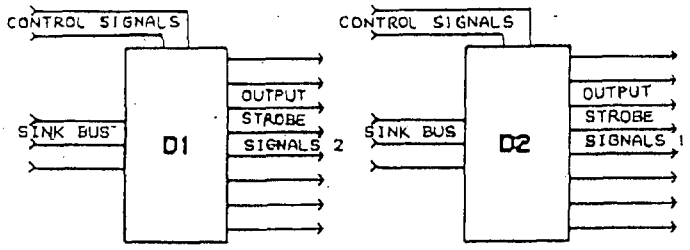


Figure 3.4.1.

A termination logic generates output strobe signals and end signals.

The first part is made of decoders which transform the sink code in output strobe signals:

- D1 generates output strobe signals for the first word and
- D2 generates output strobe signals for the second word.

The output strobe signal returns to the termination logic after it write the packet into the sink. This signal starts an output strobe signal for the second word or ends the cycle with termination signals. This part is made of three MUXes:

- MUX1 generates control signals for activation output strobe signals or for termination the cycle,
- MUX2 terminates the arbitration cycle and
- MUX3 defines how long the packet is.

If the output strobe signal does not return, the node alarms the processor.

The termination signals are used to:

- reset the synchronization logic,
- reset the arbiter,
- deactivate output strobe signals and
- delete a treated bus request.

## 3.5. TIMING

The exact logic and electrical design and the detail timing description is given in [11]. For the further experties and understanding the following is only important:

- the arbitration cycle is composed of the following sequences:
  - a packet defination,
  - a synchronization sequence,
  - an arbitration,
  - a first word transfer,
  - a changing the words on the packet bus,
  - a second word transfer and
  - a termination.
- each sequence is executed in approx. 20 ns, so one worded packets are transmitted in approx. 80 ns and two worded packets are copied in approx. 140 ns.

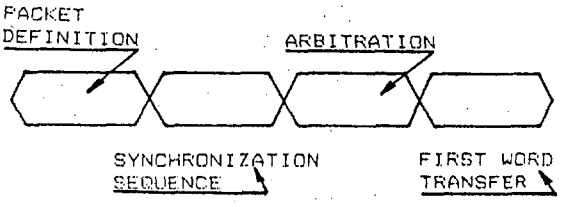


Figure 3.5.1. The first word transfer

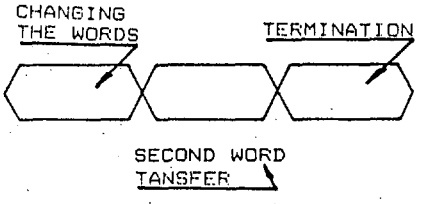


Figure 3.5.2. The second word transfer

## 4. CONCLUSION

This article explains that network's features mostly depend on the choosen architecture, used technology, internal functional organization and packet bus constructions.

The way to more efficient multiprocessor network leads to high parallel structure like cross-bar switch and fully connected networks. On the internal functional organization and bus construction field new principles must be developed to achive pipelining and interleaving in the arbiter. Faster technology brings many electronic problems which can be solved by twisted-pair cables, coaxial cables, termination circuits, on-chip connectors and a precise design. Semi or full-custom design integrited circuits will be applied to increase reliability,to protect invested knowledge and money.

Further experties on the project will give the answer how to unite this suggestions to achive high performance interconnection network.

A C K N O W L E D G M E N T

It is not just my merit that the paper has gotten this content and form. I wish to thank Peter Brajak, who has introduced me to parallel processing, has led me through the work and who has been ready to give a right advice at every time. Furthermore, I wish to thank Lojze Vogel and Marko Kovačevič for useful discussions on hardware design, as well as Dr. Sasa Prešeren, dr. A. P. Železnikar and others members of the PARSYS group.

R E F E R E N C E S

[1] Brajak P., Designing a Reconfigurable Intelligent Memory Module for performance enhancement to large scale, general purpose parallel processor, Informarika 1, Jan. 1987, page.:19-53.

[2] Hitij P., Simulator paralelnega računalnika PARSYS, Diplomsko delo, Dec. 1987, page.:12.

[3] Mackenna C., Metastabilty and Synchronization, Dec. 1987, D1-D10.

[4] Howell D., IC master, 1987, page.:2/3701.

[5] Motorola, Schottky TTL data book, 1984/85, page.:6-21.

[6] Bantz D. F., Computer system design notes, 1980, page.:I.12.

[7] Lee G., Some issues in general-purpose shared memory multiprocessing: parallelism exploitation and memory access combining, Jun. 1986, page.:82-106.

[8] Data Delay Devices, Catalog 87, 1987, page.:7, 14, 17, 36, 51.

[9] Advanced Micro Devices, Publication #08601, 1986, page.:4-99.

[10] Monolithic Memories, LSI Databook, 1987, page.:2-29, 2-30, 2-33, 2-36.

[11] Žagar A., Analiza in sinteza multiprocesorske mreže, Diplomsko delo, Sep 1988.

[12] Prešeren S., Brajak P., Vogel L., Železnikar A., A Massively Parallel Computer System Project in Yugoslavia, ISMM Inter. Conf. on Mini and Microcomputers, Florida, Dec. 1988.

[13] Brajak P., Prešeren S., Vogel L., Scheduling and Synchronization Concepts of the Parsys Parallel Processor, ISMM Inter. Conf. on Mini and Microcomputers, Florida, Dec. 1988.

[14] Siegel H.J., Interconnection Networks for Large-Scale Parallel Processing, Lexington Books, 1985

A P P E N D I X I

Simulation program for determining the number of all packets in the network (Np) after the transitional phenomenon.

```
      DEFINE THE INITIAL CONDITION
      CHOOSE A,B,C
      DO 10 I=1,10*(A+B+C+B)
      CALL SIMULATION
10    CONTINUE
      EQUATION 2.3.1.
      WRITE Np
      END
```

The core of the simulation program:

```
C..
C..  SIMULATION SUBROUTINE
C..
      SUBROUTINE SIMULATION
      INTEGER A,B,C
      COMMON A,B,C,WPGM(121),RPGM(121),
     *RPCPU(121),CPU(121),GM(121)
C..
C..  SUMMATION AND SPLITTING POINTS
C..
      GM(1)=RPGM(B+1)
      RPCPU(1)=GM(C+1)
      CPU(1)=RPCPU(B+1)+(19./20.)*CPU(A+1)
      RPGM(1)=(1./20.)*CPU(A+1)
      WPGM(1)=(1./20.)*CPU(A+1)
C..
C..  DELAYS
C..
      DO 10 K=B,1,-1
      WPGM(K+1)=WPGM(K)
      RPGM(K+1)=RPGM(K)
      RPCPU(K+1)=RPCPU(K)
10    CONTINUE
      DO 20 K=C,1,-1
      GM(K+1)=GM(K)
20    CONTINUE
      DO 30 K=A,1,-1
      CPU(K+1)=CPU(K)
30    CONTINUE
      RETURN
      END
```

A P P E N D I X I I

Program for calculating the maximum number of all packets in the network (Npm):

```
      DEFINE INITIAL CONDITION
      CHOOSE A,B,C
      Npm=0.
      DO 10 I=1,10*(A+B+C+B)
      CALL SIMULATION
      EQUATION 2.3.1.
      IF (Np.LT.Npm) GOTO 10
      Npm=Np
10    CONTINUE
      WRITE Npm
      END
```

Analiza in sinteza multiprocesorske mreže

Članek obsega pregled razvoja multiprocesorske mreže paralelnega računalnika PARSYS. Analiza mreže je podana v prvih dveh poglavjih. Kot osnova služi statistično-diskretni simulacijski model, ki se pokaže kot zelo dober pokazatelj razmer v mreži. Na teh temeljih je v tretjem poglavju zasnovana krmilna enota.