# Design and Evaluation of Communication Processors supporting Message Passing in Distributed Memory Systems

H. Corporaal J.G.E. Olk

Delft University of Technology
Mekelweg 4, P.O. Box 5031
2600 GA Delft, The Netherlands
email: heco@duteca.et.tudelft.nl eddy@duteca.et.tudelft.nl

**Abstract**

*The number of design decisions for connecting processor nodes within MIMD systems is rather large. This paper systematically introduces the most important design parameters for communication processors in MIMD systems. Together, these parameters span a multidimensional design space. Points in this space are clarified through classification of a number of existing communication processors. The design choices made for these processors are reviewed and their performance is evaluated.*

*Suitable choices of the design parameters are highly influenced by application behavior. Ideally one would like to design processors which cover a whole area in this design space. A companion paper describes a scalable and flexible design currently being realized at our laboratory.*

## 1 Introduction

Many problems require far more computing power than can be offered by single processor systems; e.g. scientific computations doing many calculations on very large data spaces, and artificial intelligence applications exploring very large search spaces. These problems drive the development of parallel computers, containing many computing nodes. In order to cooperate in the computation of a single application they must be able to communicate and synchronize with each other.

This is achieved by using either message passing, or a shared memory approach. Message passing probably offers the best prospects for massive parallel computing because of easy architectural extensibility and natural exploitation of computational and communication locality. Remember that massive parallelism results in small tasks having a high communication to processing bandwidth ratio.

Flexible and fast communication requires hardware communication support. Many designs for communication processors (CPs) are proposed and made recently [1,2,3,4,5,6,7,8,9]. The design tradeoffs made resulted in very different processors. E.g. [8] describes a very high performance but rather restrictive processor. On the other hand [4] gives a description of a CP able to support deadlock and livelock free communication for arbitrary (although degree limited) network topologies. However this processor turns out to have a very large node latency.

Based on the knowledge gained from those designs and our own design experience (see [10]), this paper investigates and evaluates the design space for hardware communication support of message passing in distributed memory computer systems. In Section 2 the most important design parameters are listed and elucidated. Together they compose a multidimensional design space for communication processors. This enables us to classify and compare existing designs. This will be done in Section 3 where different existing designs are described and characterized according to our design space model. The next section (4) evaluates the design parameters. A final section summarizes and concludes this paper. For a more extensive discussion and evaluation of the CP design space we refer to [11].

## 2 Communication processor design space

Communication processors (CPs) for MIMD systems can be designed in many different ways. In order to classify different CPs we distinguish eight important design parameters. Their names and possible values are listed in Table 1, and discussed shortly in separate subsections.

### 2.1 Network

The interconnection network of a multiprocessor system provides the processing elements of the system with a communication mechanism. For the CP it is important to know, which interconnection topologies it must handle

| design parameter | possible values |
|---|---|
| 1. Network | |
| 1. degree | fixed, variable |
| 2. topology | restricted, arbitrary |
| 3. virtual links | 1-$n$ |
| 2. Routing | |
| 1. routing path | deterministic, nondeterm. |
| 2. routing path control | DP, CP controlled |
| 3. Deadlock & livelock | |
| 1. deadlock resolvement | none, prevention, detection |
| 2. livelock resolvement | none, prevention |
| 4. Switching method | circuit, packet, wormhole switching |
| 5. Message structure | |
| 1. addressing mode | absolute, relative |
| 2. address part | fixed, varisized |
| 3. data part | fixed, varisized |
| 6. Buffering | centralized, distributed |
| 7. Interfacing | |
| 1. CP-CP interface | |
| 1. flow control | tight, loose, none |
| 2. link impl. aspects | |
| 1. dataword size | serial, parallel |
| 2. comm. protocol | synchronous, asynchronous |
| 3. control signals | encoded, decoded, none |
| 4. pin impl. | unidirectional, bidirectional |
| 2. DP-CP interface | |
| 1. flow control | |
| 1. message control | DP, CP |
| 2. field control | DP, CP |
| 2. DP view | memory, I/O, register, instr. space |
| 3. CP view | normal, special link |
| 8. Fault tolerance | none, static, dynamic |

Table 1: CP design space parameter overview.

and what degree is required. The CP can further support virtual links or virtual networks.

**Degree:** The degree of a node is defined as the number of links the node has, or the number of neighboring nodes that are directly connected. Usually CP designs only allow a fixed degree, but support for a variable degree may be present, for instance by the cascading of multiple CPs [3].

**Topology:** Many different topologies exist (see e.g. [12]). Besides the supported degree of a CP design, feasible topologies are determined by the routing support supplied by the design. The routing scheme may allow arbitrary topologies, but can also be restricted to a class of topologies. So routing may be optimized towards a certain class of topologies, e.g. k-ary n-cubes, or flexible routing is provided, for example using alterable routing tables.

**Virtual links:** If multiple connections are needed using the same physical link, the network can support virtual links. This has the additional advantage of (1) higher link utilization, because of less message blocking, and (2) the possibility of intermediate node routing. [7] and [10] describe virtual link implementations. A rather different approach to establishing virtual connections is taken in [2], where messages are divided into packets, which contain a virtual connection number. Virtual connections are not supported within the network, and therefore it lacks the corresponding advantages.

## 2.2 Routing

Since each node of a MIMD multiprocessor system can only reach a limited number of nodes directly, messages destinated for other nodes must use intermediate nodes to reach their destination. A routing algorithm is needed to decide how messages travel from their source to the destination of each message, i.e. which routing path they take and how this path is determined (see also Table 2).

| path | path control | |
|---|---|---|
| | sender | network |
| deterministic | street sign | E-cube |
| nondeterministic | random node | route always, bounded box |

Table 2: Examples of routing protocols

**Routing path control:** The routing path can either be established by the DP, i.e. the sender, or the CP, i.e. the network. When the DP specifies the routing path, the path is embedded in the message and the routing path is known when the message is sent. The path can be defined at compile-time or run-time. When the route of the path is settled in run-time, adaptive routing is possible.

If the routing path is resolved by the network, the routing algorithm selects one or more nodes, which are valid as a next node in the path to the destination. The actual forwarding of a message to such a node may depend on 'external' factors such as congestion or failing nodes.

**Routing path:** The routing path a message takes may either be deterministic or nondeterministic. Deterministic routing means that the path of a message for a given source and destination node is strictly defined and will always be the same. At each intermediate node always a fixed output link will be selected for a given destination. A deterministic routing algorithm, e.g. E-cube routing, is relatively simple but it can't react to congestion in the network or to (temporary) failure of a node.

Congestion degrades network performance. However, hot spots can be circumvented by using (random) intermediate node routing where a message is first sent (using a fixed path) to a, software determined intermediate node,

before it is routed to its real destination node. So intermediate node routing can be considered to be hybrid, since both deterministic and nondeterministic routing is possible.

Nondeterministic routing algorithms can react to certain (external) conditions, such as congestion, and thus the path of a traveling message is not always the same for a given source and destination. Nondeterministic routing is for example possible using bounded box routing [8]. However, nondeterministic routing has several disadvantages:

1. Requires more complex hardware, for instance for table lookup of possible paths and assignment of paths to messages.
2. Messages arrive out of order. This is especially cumbersome in case of packet switching. Incoming packets have to be put in the correct order for message reassembly (introducing often intolerable overhead).
3. Deadlock and livelock problems (see the next parameter in Section 2.3).

## 2.3 Deadlock & livelock

Deadlock occurs when there are cycles in the resource dependency graph. Deadlock is part of the livelock or starvation problem. A livelock situation exists when the time a process has to wait before gaining access to a resource is not bounded.

**Deadlock resolvement:** It is clear that deadlock prevention is extremely important and deadlock occurrence can't be allowed in a network, unless the deadlock probability is so low that it is negligible compared to hardware faults. There are several ways to prevent deadlock such as: class climbing [4], or the use of multiple acyclic networks [13,3]. Essentially they all map acyclic graphs on the topology in such a way that any message may reach its destination by traveling along one or more graphs. To each graph correspond resources (networks or buffers) which may only be consumed by a message in a given order.

A rather different approach is taken by the route always routing protocol. Messages are routed always in some direction even when they move further from its destination. This algorithm assumes that it is better to move messages around in the network rather than to keep them stored in a buffer. Deadlock can not occur, because only one resource is claimed at a time. However, livelock is still possible.

**Livelock resolvement:** Since livelock freedom is a stronger constraint than deadlock freedom, livelock avoidance always requires deadlock freedom. Livelock freedom in a network can, if deadlock is prevented, be established by avoiding messages to wait forever for the next buffer. This can be done by limiting the maximum time a message may occupy a buffer. This assures frequently freeing of buffer space, and by using a FIFO structure, i.e. arbitration between multiple messages for the same buffer space is fair.

This is sufficient for routing algorithms where a message always moves towards its destination. However, when messages may be routed further away from their destination, e.g. when route always is used, additional requirements are needed. The network must then guarantee message delivery and message injection[14]. It turns out that a minimal number of buffer space (measured in number of packets) is necessary, to assure that at least one message is routed in the right direction. It is clear that this contradicts unlimited message size, as allowable in wormhole routing.

## 2.4 Switching method

The switching method determines how the actual transmission of messages takes place. Alternatives are (1) circuit switching, which establishes a physical connection between source and destination, (2) packet switching, where a message is divided into packets which are sent separately, and (3) wormhole switching [1], which is essentially a combination of the former two methods.

Wormhole switching divides a message in several smaller parts, so called 'flits'. A transmission of a message then consists of a header flit, containing the destination address, followed by the flits with the message data and terminated by a trailing flit. Each node doesn't buffer a complete message but only a few flits and forwarding of a message can take place as soon as the header flit is examined. Each flit follows the same route as the header flit and because only a few flits are buffered at each node, the message is spread out over a number of nodes.

Wormhole switching is quite advantageous since it can perform both circuit and packet switching, and reduces message latency. It does not have the disadvantage of long path setup time (circuit switching) or packet reassembly (packet switching).

## 2.5 Message structure

The structure of a message defines how the data is transported over the network. It is not possible to just transport bare data over the network: so data must be packed in messages. A message usually consists of two functional parts: an address part indicating the destination, and a data part, containing the actual data. The contents of the address part depends on the applied addressing mode.

**Addressing mode:** When a node sends a message to another node it must inform the communication mechanism

to which node the message must be routed. This addressing of the destination node can be specified absolute or relative to the source node.

**Address part:** The address part may have a fixed length or may be varisized. A varisized address part allows an arbitrary number of nodes and can also be used to implement intermediate node routing [10]. The length of a varisized address part can be specified with a length field at the start of the address part, or by using a trailer, notifying the end of the address part .

**Data part:** The data part holds the data that must be carried over the network. The size of the data part, i.e. the amount of data it can hold, can either be fixed or varisized.

Fixed length data parts usually have a length of 32 to 256 bits. Using fixed length parts simplifies the buffering management, but causes internal fragmentation (partially filled data part).

A varisized data part has the advantage that, unlike fixed length parts, no additional communication overhead is introduced by transmitting empty bits. Also, there is no need to divide the message in small parts and combine these later. The length of a varisized data part can be indicated using a length field at the head of the data part or by a trailing identifier at the end. Varisized data parts may, however, complicate buffering of messages.

## 2.6 Buffering

Between the sending of a message by the source data processor and the receiving by the destination data processor, the message must be buffered in the communication network. This can be done centralized (using one shared bufferpool per node) or distributed (one buffer per physical or virtual link). Centralized buffering reduces fragmentation and therefore message blocking, however the buffer management is much more complex. Buffer access has to be serialized; this causes a severe time penalty.

## 2.7 Interfacing

The interfacing of a CP falls apart in two categories: the CP-CP interfacing, the interfacing between the CP and its adjacent CPs, and the DP-CP interfacing, the interfacing between the CP and the DP.

**CP-CP interface:** For the interfacing between two neighboring communication processors two properties must be considered: flow control, regulating the data transmission, and the implementation of the physical link, which performs the actual data transfer.

**Flow control** Since a communication processor has only limited buffer capacity, it can't always accept data coming from neighboring nodes, so some sort of flow control is needed to manage the data transferring. Flow control can be tight or loose. Tight control requires acknowledgement of each packet (or flit); an input buffer is allocated before each packet exchange. Loose control allows the sender to send packets as long as it is not informed by the receiver to stop (because of possible buffer overflow). Loose control requires far less control overhead. When using a route always protocol no flow control is necessary.

**Link implementation aspects** The implementation of the physical link is determined by (1) the dataword size, (2) the communication protocol, which can be synchronous or asynchronous, (3) flow control signals and (4) the pin implementation. Flow control signals can be encoded or decoded; in encoded control the data signals are also used for control information, decoded control uses separate pins. The pin implementation can be uni- or bidirectional.

**DP-CP interface:** This interface implements message control between CP and DP. Another aspect is called *view*: the way the CP and DP access each other.

**Message control** Both initialization and transfer of messages have to be controlled. Initialization sets up the sending or receiving of a message, and can be done by the DP or the CP. Usually this is done by the DP, but a smart CP may initiate sending/receiving of messages on its own, without consulting the DP.

The transfer of a message can be performed by either the DP or the CP. When the CP is smart, it can for example use DMA to transfer the data to/from the memory of the DP, otherwise the DP must transfer the data (software approach).

**DP view** From the DP's point of view the CP can be accessed using:

- memory space
- I/O space
- register space
- instruction space

When the CP is accessed using the memory or I/O space, the DP uses normal memory, or I/O, read/write instructions to communicate with the CP. Using these spaces for interfacing is quite universal and almost any DP can be used since there are no special requirements for the DP.

High VLSI densities make it feasible to integrate the communication processor, perhaps some memory, and the data processor onto a single chip. This allows register interfacing.

With instruction space based interfacing the instruction set of the DP is extended with special instructions for the CP. The register and instruction space interfacing techniques require that the CP and the DP are tightly coupled.

**CP view** The interface between the DP and the CP viewed from the CP can be identical to a 'normal' link, i.e. data

and control signals are the same as all other links. This makes cascading of CPs possible (e.g. for supporting higher dimensional topologies). It also may be a special DP link with interface signals fitting the DP, and having a much higher bandwidth.

### 2.8 Fault tolerance

Although it is known that fault probability in current multiprocessor systems is low, some applications may require a certain amount of fault tolerance of the system. Fault tolerance of the network can be realized statically or dynamically. Static fault tolerance means that the whole network is statically checked and routing information is updated accordingly. With dynamic fault tolerance the routing algorithm adapts to possible (temporary) failing links or nodes.

## 3 Illustrating communication processor design space

This section gives an overview of various CP designs by using the design space model of the previous section as a classification. This classification is shown in Table 3 and describes the following CPs [1]:

**Connection Machine router** [6] The CM-1 Connection Machine consists of 65,536 1-bit processors and was mainly designed as a "symbol cruncher" for artificial intelligence problems. Clumps of 16 processors are interconnected by a packet-switched 12-dimensional hypercube network[2]. The routing protocol is route always (although each CP node has some additional buffer space).

**Torus Routing Chip** [3] The TRC provides, using two virtual links per link, deadlock-free packet communications in $k$-ary $n$-cube (torus) networks. The chip was primarily intended for 2-dimensional networks, but by cascading the chips, arbitrary $n$-dimensional networks can be realized. Each node then consists of $n/2$ TRCs.

**Bounding Box Router** [8] This packet switching router is capable of routing 48-bit packets (12 bit address, 32 bit data and 4 bit reserved field) over a 64x64 grid network. The router chip lacks special interfacing to a dataprocessor, so packets must be injected into and removed from the network by using the normal links.

[1] Because of space limitations, other important designs such as described in [15] and [9] are omitted. The CPs listed here clearly illustrate the great variety of possible design choices.

[2] The processors are also connected in a 256x256 grid. However, this grid doesn't have hardware support for the routing of messages.

**DOOM Communication Processor** [4] The CP, developed for the Philips DOOM architecture, implements packet switching with packets of 256 bits (of which 10 bits for the address) to route data over an arbitrary (degree limited) network topology. A variant of class climbing is used to prevent deadlock.

**iWarp** [7] The iWarp is a single chip processor, containing a computation agent (DP) and a communication agent (CP), for high speed signal, image and scientific computing. The communication agent has four input and four output links and uses streetsign routing to transfer data over the (arbitrary) network.

**SCARCE Communication Processor** [10] This CP is part of the SCARCE[3] architecture framework for designing high performance application specific computers. The CP is currently under design, using the ASA[4] VLSI compilation system at our laboratory and is being implemented as a VLSI parametrized standard cell. Many CP parameters (like degree, nr. of virtual networks, link size and implementation) can easily be varied during design time. Adaptive routing is possible using intermediate or random node routing.

## 4 Evaluation and performance

This section suggests suitable values of the design parameters. It also discusses the achieved performance of the communication processor designs which were classified according to our design space model in the previous section.

### 4.1 Evaluation

Section 2 introduced and explained the most important design parameters for building CPs. Section 3 showed that recent developed CPs differ greatly in their choice of design parameter values. This section shortly evaluates desirable values for these design parameters based on our current design experience.

**Network:** In the past many complex network topologies are proposed for MIMD processing. This was a result from too much emphasis on the topology parameters: network diameter (directly related to node density and network exponentiality), and connectivity. A short maximum diameter was important for short message latency, a large node and arc connectivity guaranteed many independent routing paths which makes congestion control and fault tolerance (at least in principle) possible.

[3] SCalable ARChitecture Experiment.

[4] ASA is a trademark of SAGANTEC.

| **design parameter** | CM | TRC | BBR | DOOM | iWarp | SCARCE |
|---|---|---|---|---|---|---|
| 1. Network | | | | | | |
| 1. degree | fixed (12) | fixed (2) | fixed (4) | fixed (4) | fixed (4) | arbitrary |
| 2. topology | restr. | restr. | restr. | arbitr. | arbitr. | restr. |
| 2. Routing | | | | | | |
| 1. routing path | nondet. | det. | nondet. | nondet. | nondet. | nondet. |
| 2. routing path control | network | network | network | network | sender | sender, network |
| 3. Deadlock & livelock | | | | | | |
| 1. deadlock resolvement | prevention | prevention | none | prevention | none | prevention |
| 2. livelock resolvement | none | prevention | none | prevention | none | prevention |
| 4. Switching method | packet | wormhole | packet | packet | wormhole | wormhole |
| 5. Message structure | | | | | | |
| 1. addressing mode | relative | relative | relative | absolute | hybrid | relative |
| 2. address part | fixed | varisized | fixed | fixed | varisized | varisized |
| 3. data part | fixed | varisized | fixed | fixed | varisized | varisized |
| 6. Buffering | centr. | distr. | distr. | centr. | distr. | distr. |
| 7. Interfacing | | | | | | |
| 1. CP-CP interface | | | | | | |
| 1. flow control | none | tight | tight | tight | loose | loose |
| 2. link impl. aspects | | | | | | |
| 1. dataword size | serial | parallel, 8 | parallel, 12 | serial, 1 | parallel, 8 | parallel, 1–32 |
| 2. comm. protocol | synch. | asynch. | synch. | asynch. | synch. | synch. |
| 3. control signals | none | decoded | decoded | encoded | decoded | en-/decoded |
| 4. pin impl. | unidir. | unidir. | unidir. | unidir. | unidir. | bi- or unidir. |
| 2. DP-CP interface | | | | | | |
| 1. message control | | | | | | |
| 1. initialization | DP | DP | DP | DP | DP | DP |
| 2. data transfer | DP | DP | DP | DP | DP/CP | DP |
| 2. DP view | register | mem., I/O | mem., I/O | mem., I/O | register | register |
| 3. CP view | special | normal | normal | special | special | special |
| 8. Fault tolerance | none | none | none | static | none | none |

Table 3: CP design examples.

Recent research [16] showed that when wiring density has to be taken into account (which has to be done for massive parallel systems) low dimensional networks[5] favor high dimension ones. Low dimensional networks have lower message latency despite their longer average communication distance. A second, more important, argument is that those applications which are suitable for massive parallel computation (and which are luckily often those, which require the most speedup) turn out to have mostly a regular and simple process structure (like multidimensional meshes and tori). On top of that, these applications often communicate locally. Finally, massive networks have to be layout in our physical 3-d world. When high throughput, using parallel links, is required, this severly restricts the applicable topologies.

**Routing, deadlock and livelock:** A perfect router would be network (distributed) controlled, allow for nondeterministic paths to avoid hot spots, being highly fault tolerant, without having livelock problems, and always choosing the shortest path (in time). Ideally it would also have to be topology independent.

A practical hardware router must tradeoff complexity for performance. This means that the router will only support a particular (simple) class of topologies, avoiding table lookup for taking routing decisions. A restricted form of nondeterminism in the routing path is perhaps possible, although this mostly gives livelock problems (see e.g. [8]). An alternative is to use the route always protocol. This is very simple to implement, but is not livelock free, and leads to paths which can be much longer for heavy loaded networks (see e.g.[9]).

A middle of the road alternative is to use a fully deterministic, fast and livelock free routing algorithm, and to allow for software intermediate node routing. This approach is taken in [10], but has yet to be evaluated. The added advantage is that packets arrive in order, which is required in many communication situations.

**Switching method:** Most, if not all, current research projects on the design of CPs use some variant of wormhole routing because communication latency is less dependent on the number of hops (this also favors lower dimen-

[5]Dally uses multidimensional tori. Meshes may even be more beneficial since they have several advantages compared to tori. Meshes, for instance, have a smaller diameter.

sional networks). A problem with this switching method is that a link can be blocked an unlimited amount of time. There are two solutions to this problem: (1) restrict the wormsize. This approach is taken in the new transputer chip [2]). Its primary disadvantage is the necessity for packet assembly and disassembly. A far better approach (2) is to implement several virtual links per physical link. Both [7] and [10] use this approach. The number of virtual links is application dependent.

**Message structure:** The optimal message format allows for both varisized address and data parts. A varisized address part supports all dimensions within a supported class of topologies, while a varisized data part does not require packet assembly and disassembly. If (emulated) circuit switching mode is required, the data part must end with a trailer; a length field is not applicable.

The ideal addressing mode is probably relative addressing. The advantages are (1) neighbor nodes have short addresses, and (2), in regular communication patterns the address calculation can be done at compile-time, and is equal for all nodes.

**Buffering:** Although centralized buffers are more economical used, distributed buffering is preferred, because (1) buffer control is much easier (and therefore faster to implement) and (2), the deadlock problem is reduced. E.g. if two connected nodes communicate which each other, they use different buffers, and cannot deadlock each other (at least not for this reason).

The size of the buffers greatly depends on the communication pattern and routing algorithm. Larger buffers reduce the amount of misrouting in adaptive routing protocols, however the response to congestion will be delayed. This is an interesting research area. Large buffers between DP and CP are necessary to tolerate the speed differences between inter CP links and the CP-DP connection.

**Interfacing:** The interfacing method is highly dependent on the type of applications. The CP-CP interface should be parallel (limited by pincount), and bidirectional (unless the communication pattern is unidirectional). Although a bidirectional protocol is somewhat more complex, it (almost) doubles the possible dataword size. The flow control is loose and receiver initiated. In case of a high degree, the control signals should be shared.

A register mapped (on chip CP) DP-CP interface is best suited for fine grain, high performance communication, but is of course impossible for off the shelf CPUs. Off chip CPs should have a special (high speed) DP connection (e.g. memory mapped), and at least support some form of DMA.

**Fault tolerance:** Fault tolerance is not that important as people thought in history. The failure rate of hardware in parallel systems is less dramatic than expected. Nobody implemented dynamic fault tolerance. [4] implements static fault tolerance, but their routing protocol uses table lookup, which is to expensive. The route always protocol is probably best suited for dynamic fault tolerance, but we have discussed its problems. Intermediate node routing makes (at least in principle) static fault tolerance possible.

## 4.2 Performance

Table 4 shows the evaluated performance parameters for the various designs. The evaluated parameters are:

- total number of link pins (excluding possible DP interface links), including control signals.
- pin usage per link, number of pins for a single input link (including possible control signals.
- total throughput of all links (both input and output).
- throughput of a single link.
- throughput of single linkpin (corrected for possible control pins).
- pin throughput per cycle: normalized throughput, according to the internal CP clock cycle
- throughput efficiency: throughput for pure data only, not counting control transfers.
- minimal hop latency: additional time delay for each hop a message takes.

All designs use unidirectional linkpins. From this table we can draw several conclusions:

1. BBR & iWarp show how to achieve high throughput; use of (1) a high pin count and (2) a high pin throughput per cycle which is realized by using a higher external clock frequency. They further use simple routing algorithms, which results into low latency.
2. The hop latency of DOOM is very large due to a complex livelock free routing protocol and the use of centralized buffers.
3. CM uses its links rather inefficiently (1 dimension at a time). Many CM applications therefore do not use the Hypercube network, but use the grid network (without routing support).

The effective throughput and latency of these CPs highly depends on the communication structure of application programs. Blocking of messages may reduce the theoretically attainable performance drastically, especially when wormhole routing is used without adequate internal buffer space. Using virtual links (as in iWarp and SCARCE) will certainly reduce blocking.

| **evaluation parameter** | CM | TRC | BBR | DOOM | iWarp | SCARCE[a] |
|---|---|---|---|---|---|---|
| total number of linkpins | 24 | 48 | 112 | 8 | 104 | 72 |
| pin usage per link | 1 | 12 | 14 | 1 | 13 | 9 |
| total throughput (Mbit/s) | 2.67 | 128 | 3840 | 80 | 2560 | 1280 |
| throughput per link (Mbit/s) | 1.33[b] | 32 | 480 | 10 | 320 | 160 |
| throughput per linkpin (Mbit/s) | 1.33[b] | 2.67 | 34.3 | 10 | 24.6 | 17.8 |
| pin throughput per cycle (bit) | 0.33[b] | 0.67 | 3.43 | 1 | 2 | 1 |
| throughput efficiency | 8.33% | 67% | 91.7% | $< 80\%$ | 61.5% | 88.9% |
| minimal hop latency | 48$\mu$s | 250ns | 100ns | 25,6$\mu$s | 100-150 ns | 50ns |

[a] Implementation using 8-bit wide internal datapath, eight 8-bit physical links with encoded control. Expected clock frequency is 20MHz.
[b] Peak performance, link is not used continuously but only 1/12 of the time.

Table 4: CP performance evaluation.

## 5 Conclusions

This paper investigated the design space of communication processors supporting message passing in distributed memory computers by elucidating the most important design parameters: These parameters form a multidimensional design space. Points in this space were illustrated using a classification, comparison and evaluation of several existing CP designs and the SCARCE CP [10] currently being designed at our laboratory.

The design space model discussed in this paper was of great value during the construction of our scalable and flexible SCARCE CP design, which covers a whole design subspace. We think that this design overview is an invaluable tool for every message passing MIMD designer. The use of our model and suggested design parameters will simplify location and implementation of points in the multidimensional design space which are not yet explored.

Keys to high performance CPs are, simple routing strategies, high pincounts (parallel datapaths), high link frequencies (possibly higher than internal clockrates!), bidirectional datapaths (which are underexplored yet), reducing blocking through virtual links, adequate internal buffer sizes and simple compression schemes.

## References


[1] Charles L. Seitz et al. *The Hypercube Communications Chip*. Technical Report, Dept. of Computer Science, California Institute of Technology, 1985.

[2] Dick Pountain. Virtual channels: the next generation of transputers. *BYTE Magazine (International Edition)*, 15(4):E&W 3–12, April 1990.

[3] William J. Dally. *A VLSI Architecture for concurrent Data Structures*. Kluwer Academic Publishers, 1987.

[4] J. K. Annot and R. A. H. van Twist. A novel deadlock free and starvation free packet switching communication processor. In *Parallel Architectures and Languages Europe*, page 68, 1987.

[5] Daniel A. Reed and Dirk C. Gronwald. The performance of multicomputer interconnection networks. *IEEE computer*, June 1987.

[6] Daniel Hillis. *The Connection Machine*. MIT Press, 1985.

[7] e.a. Borkar S. Iwarp: an integrated solution to high-speed parallel computing. In *Proceedings of Supercomputing '88*, 1988.

[8] W.G.P. Mooij. *Packet Switched Communication Networks for Multi-Processor Systems*. PhD thesis, University of Amsterdam, 1989.

[9] C. Germain et al. A new communication design for massively parallel message passing computers. In *IFIP Working Congerence on Decentralized Systems*, Lyon, December 1989.

[10] H. Corporaal and J.G.E. Olk. A Scalable VLSI MIMD Routing Cell. In *DMCC-6 conference proceedings, Portland*, April 1991.

[11] H. Corporaal and J.G.E. Olk. *Designing Communication Processors for message passing distributed systems*. Technical Report 1-68340-44(1991)04, Delft University of Technology, Depart. of EE, May 1991.

[12] Ferdinand Peper. *Efficient Network Topologies For Extensible Massively Parallel Computers*. PhD thesis, Technical University Delft, 1989.

[13] Jesshope, Miller, and Yantchev. High performance communications in processor networks. In *Computer Architecture Conference Proceedings*, 1989.

[14] John Y. Ngai. *A Framework for Adaptive Routing in Multicomputer Networks*. PhD thesis, California Institute of Technology, 1989. CS-TR-89-09.

[15] Daniel A. Reed and Richard M. Fujimoto. *Multicomputer Networks Message-Based Parallel Processing*. MIT press, 1987.

[16] William J. Dally. Performance analysis of k-ary n-cube interconnection networks. *IEEE transactions on Computers*, 39(6):775–785, June 1990.