A NEW HIERARCHICAL INTERCONNECTION NETWORK
AND ITS PERFORMANCE ANALYSIS

L. Tissa Samaratunga and Syed Masud Mahmud
Department of Electrical and Computer Engg., Wayne State University, Detroit, Michigan 48202.

Abstract - In this paper we propose a new hierarchical interconnection network (HIN) which can be used to build large cluster-based systems. Our HIN is an evolved version of the HIN presented in [6]. The proposed HIN offers a number of advantages: increased fault tolerance; better performance; and, ability to be used in a cache based system. An analytical model has been developed to measure the performance of the proposed network. The analytical model has been developed based on the assumption that “the requests which are not accepted during a cycle are blocked and resubmitted during the next memory cycle”, in contrast to the commonly made assumption that “the requests which are not accepted during a cycle are discarded”. The performance of the proposed HIN is measured in terms of memory bandwidth of the system. For the same set of system parameters and the same set of assumptions for the analytical models, we compared the performance of our system with the system presented in [6]. Our system gives significantly better performance than the system of [6].


I. INTRODUCTION

Cluster-based multiprocessor systems are very attractive since they need less expensive interconnection networks compared to the noncluster-based systems. However, the success of a cluster-based system depends on the locality of references. Generally a hierarchical interconnection network (HIN) is used to build a cluster-based multiprocessor system with a large number of processing elements and memory modules.

Several HINs are presented in the literature [6]-[10]. A new HIN is presented in this paper. Our HIN is an improved version of the one presented in [6]. The HIN presented in [6] is built using a number of small crossbar switches and a number of hierarchy of buses. All local requests are satisfied by local crossbar switches. Nonlocal requests are satisfied by nonlocal crossbar switches and by the hierarchy of buses. The organization of a local level cluster is shown in Figure 1.

The interconnection within clusters is provided by employing small crossbar switches. Each crossbar switch has two types of inlets and outlets. A local level crossbar has (n1 + b1) inlets and (m1 + a1) outlets. n1 and b1 inlets are connected to the n1 processors and the b1 down words outlets of the 1st. level crossbar, respectively, whereas the m1 outlets are connected to the m1 memory modules and the a1 outlets are connected to the a1 inlets of the first crossbar.

All the ith nonlocal level (ie. between local and global levels) crossbars have (k1 * a1 + bi+1) inputs and (k1 * b1 + ai+1) number of outputs.

This system was developed by alleviating or eliminating the drawbacks of the system proposed in [6]. Several features of the proposed system in contrast to the system presented in [6] are noteworthy: downward traffic is supported by the same crossbar switch network that caters to the upward traffic; the crossbar switches at two consecutive levels are connected by multiple channels. This adds two desirable features: increased performance and more flexibility. In a cache based system, since the system uses the same network for the upgoing and downgoing traffic, switching elements can be exploited to route the traffic so that the total cache-coherence traffic intensity throughout the system is minimized.

II. DESCRIPTION OF THE HIERARCHICAL SYSTEM ARCHITECTURE

The m - level hierarchical system architecture presented in this paper is an evolved version of the architecture presented in [6]. This architecture consists of m-levels of hierarchy (Figure 1 depicts the architecture of a 3 - level hierarchical interconnection network). The processors and memory modules are grouped into local clusters. A group of these local clusters are used to form first level clusters. These first level clusters are grouped to form second level clusters. This process of formation of clusters is continued until the system builds up to a hierarchy of m levels. The highest level is called the global level, and the lowest level is called the local level, where as all the intermediate levels are called non-local levels [6].

The total number of processors is given by N = n1 . k1 . k2 . . . km .

2.k and the total number of memory modules in the system is given by

M = m1 . k1 . k2 . . . km-2.k

where as N1 is the number of processors per local cluster and m1 is the number of memory modules per local cluster. The organization of a local level cluster is shown in Figure 2.

There are k1 number of local level clusters under a first level cluster, and there are k2 number of first level clusters under a second level cluster and so on. Likewise there are k number of clusters forming under the global level cluster.

The interconnection within clusters is provided by employing small crossbar switches. Each crossbar switch has two types of inlets and outlets. A local level crossbar has (n1 + b1) inlets and (m1 + a1) outlets. n1 and b1 inlets are connected to the n1 processors and the b1 down words outlets of the 1st. level crossbar, respectively, whereas the m1 outlets are connected to the m1 memory modules and the a1 outlets are connected to the a1 inlets of the first crossbar.

All the ith nonlocal level (ie. between local and global levels) crossbars have (k1 * a1 + bi+1) inputs and (k1 * b1 + ai+1) number of outputs.

This system was developed by alleviating or eliminating the drawbacks of the system proposed in [6]. Several features of the proposed system in contrast to the system presented in [6] are noteworthy: downward traffic is supported by the same crossbar switch network that caters to the upward traffic; the crossbar switches at two consecutive levels are connected by multiple channels. This adds two desirable features: increased performance and more flexibility. In a cache based system, since the system uses the same network for the upgoing and downgoing traffic, switching elements can be exploited to route the traffic so that the total cache-coherence traffic intensity throughout the system is minimized.
Notations:
The notations used in this analysis are as follows:
- \( N \): total number of processors
- \( M \): total number of memory modules
- \( n_i \): number of processors in a local cluster
- \( m_i \): number of memory modules in a local cluster
- \( k_i \): number of (i-1)th level clusters in an ith level cluster.
- \( a_i \): number of input lines to the ith level crossbar from an
  (i-1)th level crossbar.
- \( b_i \): number of output lines from the ith level crossbar to
  the (i-1)th level crossbar.
- \( s_i \): probability that a processor's request is of ith level
- \( P_{ext} \): probability that a processor addresses a memory module
  outside its own cluster
- \( pG \): rate of request at an input of the GIN
- \( pN_i \): rate of request at an input of an ith level incoming
  request from the lower level.
- \( u_i \): portion of \( pN_i \) that goes to the ith local level
- \( R_i \): average number of requests for a downgoing outlet of an
  ith level crossbar.
- \( P_{global} \): rate of request at an output of the GIN
- \( P_{incluster} \): rate of request at a memory module due to requests
  from the processors which reside in the same cluster.
- \( P_{outcluster} \): rate of request at a memory module due to requests
  coming through CIN
- \( P_m \): total rate of request at a memory module.
- \( BW_m \): bandwidth of the m-level multi processor system.
- \( r_{ij} \): probability that an ith level request is accepted by
  a jth level crossbar due to a request generated by an
  (i-1)th level downgoing outlet during an ith level cycle.
- \( pA_{ij} \): rate of request at an input of an ith level crossbar
  coming through CIN
- \( BW_{mi} \): memory bandwidth due to an ith local level only.

III. PERFORMANCE ANALYSIS
The model presented in this paper is developed by making the
following assumptions:
1) The multiprocessor system is synchronous with \( N \)
   processors and \( M \) memory modules.
2) The request generated in a cycle is random and are
   independent of each other.
3) The requests which are not accepted during a cycle are queued
   to the appropriate memory module and are resubmitted for
   the same memory modules in the next cycle.
4) \( \Psi \) is the probability with which a processor generates a request
   in a cycle.
Assumption (3) makes the model very much more realistic compared
with the models often developed with the assumptions that the requests
which are not accepted during a memory cycle are discarded and the requests
issued in successive cycles are independent of the requests issued in the previous cycle.

The Analytical model consists of three stages. At the first stage a
probabilistic model is used to analyze the performance of the system
using the assumptions that the requests which are not accepted during
a memory cycle are discarded and the requests issued in successive cycles are independent of the requests issued in the previous cycle. Therefore, all
the processors are qualified to make new requests in every memory cycle. But in reality, the requests which are not satisfied in a certain memory cycle are queued at various stages of the network, and consequently the corresponding processors are disabled to make new requests until the requests they have already made are satisfied.

The second and the third stages of the model are developed in order
to include the bandwidth contributions made by the requests of various levels. At this stage an iterative technique is used to calculate the bandwidth components
precisely, for a given fraction of active processors. Finally, at the third stage a Steady State Flow Analysis is used to evaluate the
bandwidth of the system for a given request pattern, according to the assumption that the requests which are not accepted during a cycle are queued to the appropriate memory module and they are resubmitted for the same memory module during the next memory cycle.

A. Stage I: Performance Analysis Using Probabilistic Model:
At this first stage the performance of the system is evaluated using
the assumptions that the requests which are not accepted during
a memory cycle are discarded, and the requests issued in successive
cycles are independent of the requests issued in the previous cycle. Let \( \Psi \) be the probability that a processor needs to generate a request in a cycle. The probability that a processor addresses a memory module outside its own cluster is:
\[ P_{ext} = (1-\Psi) \]
(1)
Let \( q_{io} \) be the probability that there is at least one outcluster request
at the ith level crossbar for one of the out cluster memory modules. Therefore \( PN_i \) can be calculated as:
\[ PN_i = q_{io} \cdot \frac{(M-m_i)}{a_i} \]
if \( q_{io} \cdot (M-m_i) < a_i \)
(2)
else
\[ PN_i = 1 \]
Similarly it can be shown that, at the ith level crossbar:
\[ q_{io} = 1 - \prod_{k=1}^{(M-m_i)} \left(1 - \frac{\Psi}{a_i} \right) \]
(3)
\[ PN_i = q_{io} \cdot \frac{(M-m_i)}{a_i} \]
if \( q_{io} \cdot (M-m_i) < a_i \)
(4)
Let \( P_{in} \) be the average number of requests for a downgoing outlet of an ith level crossbar.
\[ P_{in} = P_{global} \cdot BW_{mi} \]
for the other levels can be calculated as follows:
At (i-1)th level, in order to calculate \( P_{in} \), we have to consider
the requests coming to the (i-2)th level crossbar from both higher levels
and lower levels. The requests which are competing for the downgoing
outlets of the ith level crossbar are i level (incluster) requests and higher level (outcluster) requests. Let \( P_{out} \) be the probability that there is no outcluster requests at an ith level crossbar, requesting a given memory module within the cluster and, let \( P_{in} \) be the probability that there is no such incluster requests. Also let \( R_i \) denotes the average
number of requests requesting a downgoing outlet of an ith level crossbar.
\[ R_i = (1 - \Psi_i \cdot P_{out}) \cdot m_i \]
(5)
\[ P_{in} = R_i \cdot B_i \]
if \( R_i < B_i \)
(6)
\[ P_{in} = 1 \]
otherwise.
\[ R_i \] is the probability that there is a request for a given memory
module. Therefore, the total bandwidth of the system is given by:
\[ BW = RL \cdot M \]
B. Stage 2: Decomposition of the Probabilistic Model:

At this stage we compute the constituents of the overall bandwidth. The total bandwidth is composed of the probabilistic model of each level by decomposing the probabilistic model presented in stage 1. The decomposition of the probabilistic model is necessary to apply the steady state flow analysis for keeping track of the number of processors which are blocked and waiting for service from networks at different levels.

Contribution to the Memory Bandwidth by Different Types of Requests

1) contribution by the local requests:

Let \( P_{m,1} \) be the probability that a local request is accepted by a memory module during a memory cycle. Assuming that requests for memory are granted on a fair basis, then,

\[
P_{m,1} = \frac{P_{m,1}}{P_{m,1} + P_{m,2}}
\]

Then, the bandwidth due to local requests can be computed as:

\[
BW_{m} = P_{m,1} \cdot M
\]

2) contribution by the nonlocal requests:

The average number of requests coming from the upper level is \( P_{b,i} \). Let the average number of requests coming from the lower levels which are directed to the downward outlets be \( R_{local,i} \).

\[
P_{A,i} = (1 - P_{m,i}) \cdot m_i \cdot k_i \cdot k_{i-1}
\]

Using the above expression iteratively, the probability that an \( i \) th nonlocal request is accepted by a memory module \( P_{m,i} \) can be calculated. Then the memory bandwidth due to \( i \) th nonlocal requests is:

\[
BW_{m,i} = P_{m,i} \cdot M
\]

3) Contribution by Global Requests:

Using the same method above, the probability that a global request is accepted by a memory module during a memory cycle, \( P_{m,g} \) can be calculated. Then the total memory bandwidth due to global requests is:

\[
BW_{m,g} = P_{m,g} \cdot M
\]

Therefore, the total memory bandwidth of the system is:

\[
BW = BW_{m} + \sum_{i=1}^{m-2} BW_{m,i} + BW_{m,g}
\]

C. Stage 3: Steady State Flow Analysis for the Decomposed Models:

Forgoing stages of the analytical model is based on the assumptions that the requests which are not accepted during a memory cycle are discarded and the requests issued in successive cycles are independent of the requests issued in the previous cycle. Although the error introduced by this assumption is not significant in case of simple crossbar and multiple-bus systems [1]-[4], that for the systems based on HIN can be substantial depending upon the distribution of the memory references.

At this stage the correction is made to the above analysis by considering that the rejected requests are resubmitted in the next memory cycle. The decomposed models can be used to keep track of the fractions of processors which are blocked and waiting for service from the networks at different levels. Using the decomposed model, initially we compute the bandwidth contribution by different types of requests for a given set of values of the parameters \( R_b, L_2, s_2, t_2, s_2 \). By comparing the total number of requests submitted of a certain type, say local, with the bandwidth contribution of that type of request we determine the number of processors that are waiting for service at the corresponding level. After knowing the fractions of the processors which are blocked at different levels, we can update the rate requests for different levels, and then recompute the bandwidth contribution of every type of request. This process is repeated until a steady state is reached. At this point the total memory bandwidth will be the actual bandwidth of the system. Let \( f \) be the fraction of the processors which are active at the steady state and let \( q \) be the probability that a memory is requested by at least one active or queued processor.

As per [9], this is calculated to be:

\[
q = 1 - \left( 1 - s_1 \right)^N M
\]

Now the value of \( q \) can be used to determine the value of the bandwidth of the system. Let \( f_L, f_i, f_g \) be the fractions of processors blocked for local, \( i \) th nonlocal and global memory modules respectively. Let \( f_{ext} \) be the fraction of processors that are blocked for the memory modules which are outside the processors' own cluster. Now, the probability that a processor is either generating a new request for an external memory module or it is already blocked for such a memory module is:

\[
P_{ext} = (1 - s_1) f_L + f_{ext}
\]

\[
l_i = \frac{f_{ext} + f_i}{\sum_{j=1}^{m-2} (f_{ext} + f_g)}
\]

Using the adjusted values of \( P_{ext} \) and \( l_i \) we compute the values of \( P_i, P_g, P_{global}, P_i, \) and \( R_{local,i} \) using equations (2) - (6). Then the values of \( BW_{m,i}, BW_{m,g} \) are determined using (8) (11) and (12). These bandwidth contributions by different types of requests are then used to adjust the values of \( f_L, f_i, \) and \( f_g \).

For a given value of \( f \) we can iteratively determine the value of \( f_L, f_i, \) and \( f_g \) using the following recursive equations:

\[
f_L(k+1) = f_{ext} + f_i(k) - \frac{BW_{m,i}}{N} \quad 0 \leq i \leq m-1
\]

The algorithms for the calculation of bandwidth:

**ALGORITHM 1**

1. For a given value of \( f \), determine \( f_L(0), f_i(0) \) and \( f_g(0) \) for \( 1 \leq i \leq m-2 \).

2. \( n = 1 \)

**REPEAT**

3. \( n := n + 1 \)

4. Determine \( P_{ext} \) and \( l_i, \quad 1 \leq i \leq m-2 \) using (15) - (16).

5. Do all the analysis shown by (2) through (13) and get \( BW_{m,i}, BW_{m,g} \) and \( BW \) from (8), (11) and (12).

6. Determine \( f_L(n+1), f_i(n+1) \) for \( 1 \leq i \leq m-2 \), and \( f_g(n+1) \) using (17).

7. Error = \( |f_L(n+1) - f_L(n)| + 1 + f_g(n+1) - f_g(n)| \)

\[\text{For } i = 1 \text{ to } m-2 \text{ do}
\]\n
\[\text{Error := Error + } |f_L(n+1) - f_L(n)| \]

\[\text{UNTIL Error < } \varepsilon \quad (\text{where } \varepsilon \text{ is a very small number})
\]

8. Determine the total memory bandwidth of the system as:

\[
BW = BW_{m,i} + BW_{m,g} + BW_{m,i} + \ldots + BW_{m,m-2} + BW_{m,g}
\]

The bandwidth of the system at steady state is \( BW \). Therefore, we use Algorithm-2 to satisfy the condition:

\[
BW = fN \psi
\]

**ALGORITHM 2**

1. \( f = 1.0; \text{Done} := \text{False}; \)

**REPEAT**

2. Use the Algorithm-1 to determine \( BW \)

3. If \( |BW - fN \psi| < \varepsilon \) (where \( \varepsilon \) is a very small number)

\[\text{Then Done := True}
\]

Else determine the value of \( f \) for the next iteration using Newton-Rapson Method.

**UNTIL** Done.

4. Accept \( BW \) as the bandwidth of the system.
IV. NUMERICAL RESULTS AND DISCUSSIONS

The performance of our HIN with two link per channel is presented in Tables I-IV. The performance of the HIN of [6] is also presented in these tables to compare this system with our system. (HIN-1 represents the HIN of [6] and HIN-2 represents our system). The performance is shown for two different reference patterns: hierarchical non-uniform (HNR) and uniform (UR) reference patterns. From these tables it is seen that for HNR patterns the performance of our HIN is significantly better than the HIN of [6]. Also, it is shown that the UR pattern the performance of our system is 53 – 220% better than the other system.

<table>
<thead>
<tr>
<th>Total number of Processors &amp; Memory Modules (N x M)</th>
<th>HIN-1</th>
<th>HIN-2</th>
<th>HIN-1</th>
<th>HIN-2</th>
</tr>
</thead>
<tbody>
<tr>
<td>16</td>
<td>10.5</td>
<td>5.86</td>
<td>11.85</td>
<td>8.07</td>
</tr>
<tr>
<td>24</td>
<td>30.0</td>
<td>9.39</td>
<td>31.11</td>
<td>11.44</td>
</tr>
<tr>
<td>32</td>
<td>19.23</td>
<td>6.96</td>
<td>22.42</td>
<td>12.72</td>
</tr>
<tr>
<td>48</td>
<td>22.8</td>
<td>9.98</td>
<td>27.75</td>
<td>13.59</td>
</tr>
<tr>
<td>64</td>
<td>25.6</td>
<td>9.99</td>
<td>35.07</td>
<td>14.23</td>
</tr>
<tr>
<td>56</td>
<td>27.35</td>
<td>5.99</td>
<td>38.4</td>
<td>14.72</td>
</tr>
<tr>
<td>64</td>
<td>28.24</td>
<td>5.00</td>
<td>43.74</td>
<td>15.10</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Total number of Processors &amp; Memory Modules (N x M)</th>
<th>HIN-1</th>
<th>HIN-2</th>
<th>HIN-1</th>
<th>HIN-2</th>
</tr>
</thead>
<tbody>
<tr>
<td>32</td>
<td>21.04</td>
<td>5.91</td>
<td>22.67</td>
<td>11.38</td>
</tr>
<tr>
<td>48</td>
<td>31.25</td>
<td>5.95</td>
<td>33.27</td>
<td>12.11</td>
</tr>
<tr>
<td>64</td>
<td>41.25</td>
<td>9.98</td>
<td>43.45</td>
<td>14.16</td>
</tr>
<tr>
<td>80</td>
<td>50.85</td>
<td>6.00</td>
<td>53.24</td>
<td>14.86</td>
</tr>
<tr>
<td>96</td>
<td>56.95</td>
<td>5.99</td>
<td>63.07</td>
<td>15.36</td>
</tr>
<tr>
<td>112</td>
<td>66.97</td>
<td>6.01</td>
<td>71.72</td>
<td>15.73</td>
</tr>
<tr>
<td>128</td>
<td>67.52</td>
<td>6.02</td>
<td>82.90</td>
<td>16.07</td>
</tr>
<tr>
<td>144</td>
<td>73.58</td>
<td>6.00</td>
<td>91.09</td>
<td>16.25</td>
</tr>
<tr>
<td>160</td>
<td>74.57</td>
<td>6.04</td>
<td>101.57</td>
<td>16.64</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Total number of Processors &amp; Memory Modules (N x M)</th>
<th>HIN-1</th>
<th>HIN-2</th>
<th>HIN-1</th>
<th>HIN-2</th>
</tr>
</thead>
<tbody>
<tr>
<td>256</td>
<td>22.35</td>
<td>6.09</td>
<td>89.35</td>
<td>17.11</td>
</tr>
<tr>
<td>512</td>
<td>35.98</td>
<td>6.25</td>
<td>97.74</td>
<td>17.79</td>
</tr>
<tr>
<td>768</td>
<td>34.90</td>
<td>6.58</td>
<td>100.28</td>
<td>17.91</td>
</tr>
<tr>
<td>1024</td>
<td>35.59</td>
<td>6.29</td>
<td>101.72</td>
<td>18.03</td>
</tr>
<tr>
<td>1280</td>
<td>35.61</td>
<td>6.29</td>
<td>102.35</td>
<td>18.10</td>
</tr>
<tr>
<td>1536</td>
<td>37.38</td>
<td>6.29</td>
<td>103.04</td>
<td>18.16</td>
</tr>
<tr>
<td>1792</td>
<td>37.62</td>
<td>6.29</td>
<td>103.46</td>
<td>18.20</td>
</tr>
<tr>
<td>2048</td>
<td>37.67</td>
<td>6.29</td>
<td>103.66</td>
<td>18.21</td>
</tr>
<tr>
<td>2304</td>
<td>37.63</td>
<td>6.29</td>
<td>103.83</td>
<td>18.22</td>
</tr>
<tr>
<td>2560</td>
<td>37.75</td>
<td>6.29</td>
<td>103.96</td>
<td>18.22</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Total number of Processors &amp; Memory Modules (N x M)</th>
<th>HIN-1</th>
<th>HIN-2</th>
<th>HIN-1</th>
<th>HIN-2</th>
</tr>
</thead>
<tbody>
<tr>
<td>256</td>
<td>82.22</td>
<td>6.09</td>
<td>172.16</td>
<td>16.94</td>
</tr>
<tr>
<td>512</td>
<td>91.75</td>
<td>6.22</td>
<td>250.00</td>
<td>17.76</td>
</tr>
<tr>
<td>768</td>
<td>95.41</td>
<td>6.34</td>
<td>268.30</td>
<td>18.18</td>
</tr>
<tr>
<td>1024</td>
<td>97.46</td>
<td>6.33</td>
<td>273.24</td>
<td>18.22</td>
</tr>
<tr>
<td>1280</td>
<td>101.53</td>
<td>6.33</td>
<td>277.30</td>
<td>18.25</td>
</tr>
<tr>
<td>1536</td>
<td>104.65</td>
<td>6.33</td>
<td>279.94</td>
<td>18.27</td>
</tr>
<tr>
<td>1792</td>
<td>106.07</td>
<td>6.33</td>
<td>280.98</td>
<td>18.29</td>
</tr>
<tr>
<td>2048</td>
<td>107.59</td>
<td>6.33</td>
<td>280.99</td>
<td>18.30</td>
</tr>
<tr>
<td>2304</td>
<td>108.85</td>
<td>6.33</td>
<td>280.82</td>
<td>18.27</td>
</tr>
<tr>
<td>2560</td>
<td>110.57</td>
<td>6.33</td>
<td>279.93</td>
<td>18.29</td>
</tr>
</tbody>
</table>

IV. CONCLUSION

Hierarchical Interconnection Networks provide a very much cost effective means of interconnection among a large number of processors, when the request pattern of the processors is of hierarchical nature. Mahgoub and Elmagarmid [6] proposed a cluster-based system which is based on a generalized hierarchical interconnection network. A number of drawbacks of this system are recognized. In this paper, a new Hierarchical Interconnection Network is proposed. This is an improved version of the system proposed in [6] and offers more advantages: increased fault tolerance; increased adaptability to a given application; and, ability to be used in a cache based system. An accurate analytical model is developed in order to assess the performance of the synchronous system. This model is based on the assumption that "the requests which are not accepted during a cycle are queued to the appropriate memory module; i.e. they are re-submitted for the same memory modules during the next memory cycle", in contrast to the commonly made assumption that "the requests which are not accepted during a memory cycle are discarded and the requests issued in successive cycles are independent of requests issued in the previous cycle". In case of HIN based systems. The results of the analytical model are used to demonstrate the bandwidth characteristics of the system. Bandwidth of the system is very sensitive to the request pattern and to the widths of the channels provided at different levels of the system. The results of our analysis show that our system performs significantly better compared to the system proposed in [6]. By studying the system characteristics as shown in this paper, a HIN based system can be designed to meet the best requirements of a given application, so that the most cost effective use is made of the interconnection network.

References