Computing and Informatics, Vol. 23, 2004, 415–435

PERFORMANCE EVALUATION AND IMPLEMENTATION OF TWO ADAPTIVE ROUTING ALGORITHMS FOR XGFT NETWORKS

Heikki Kariniemi, Jari Nurmi

Institute of Digital and Computer Systems
Tampere University of Technology
P. O. BOX 553, 33101 Tampere, Finland
e-mail: {heikki.kariniemi, jari.nurmi}@tut.fi

Manuscript received 22 December 2004

Abstract. EXtended Generalized Fat Trees (XGFT) are Bidirectional Multistage Interconnection Networks (BMIN). They are more scalable for different system sizes and different performance requirements than fat trees from which they have evolved. The improved scalability has been achieved by allowing switches with different number of ports to be used in different switch stages of these hierarchical networks. XGFTs can be constructed from two separate networks for routing packets upwards and downwards in the XGFT. These up-routing and down-routing networks can be implemented separately with small switches which are connected to each other within the switch nodes of the XGFT. This kind of XGFT achieves higher performance if its topmost root switches are connected to each other with additional links, and if adaptive Turn-Back-When-Possible (TBWP) routing algorithm is used instead of shortest-path routing algorithms. This paper shows that the TBWP has always simple and feasible hardware implementations independently of the structure of the XGFT. This is achieved by address space encoding which eliminates complex computations from the routing decision functions. This paper presents also a new shortest-path routing algorithm named Turn-Back (TB). The TB algorithm was designed for such XGFT implementations where the up-routing and down-routing of the packets is performed with one larger switch block within the switch nodes, and where shortest-path routing produces good performance. It is shown in this paper that the TBWP and TB route packets correctly to their destinations. In addition, the performances of the routing algorithms are evaluated with simulations and compared. Simulation results show that the TB is able to produce higher performance than the TBWP with different traffic patterns. They also show that the performance of the XGFTs could be improved by suitable mapping of the communicating processes to the processor leaf nodes.

Keywords: eXtended Generalized Fat Tree (XGFT), top-down recursive generation of the XGFTs, address space encoding, adaptive Turn-Back-When-Possible (TBWP) routing algorithm, adaptive Turn-Back (TB) shortest-path routing algorithm, Network-On-Chip implementations, performance simulations

1 INTRODUCTION

Platform-based design flows are becoming commonly used in System-on-Chip (SoC) circuit design [1, 2, 3], and an increasing number of the functions of the SoC circuits will be implemented with software [4]. For this reason, future SoC circuits will contain several programmable processors and reconfigurable logic blocks so that they could be flexibly modified for different applications. As a consequence of this, their communication infrastructures should be suitable for different traffic patterns produced by various applications, which can be modeled as e.g. Kahn process networks (KPN) [5, 6] mapped to computing resources. The applications could be e.g. wireless communication, voice and video compression and decompression, and computer graphics.

If Network-On-Chip (NOC) communication infrastructure would be designed for a group of some particular applications, its routing resources and capacity could be dimensioned more optimally to fulfill the requirements, and the network topology would become irregular [7]. However, it is not always necessary to do static allocation of computing resources to application processes, if programmable and reconfigurable general-purpose computing platforms are used. In addition, dynamic run-time allocation of computing resources for processes would make it possible to run multiple applications on platforms of limited sizes while it would also allow high utilization of computing resources. Because standards and applications may also change in short terms, programmability, reconfigurability, and general-purposeness will also be required properties. For this reason also, the communication infrastructure should be flexibly configurable so that it would be able to produce sufficient performance for various platform configurations with different mappings of the application processes to computing resources.

EXtended Generalized Fat Trees (XGFT) [8, 9], which have been developed from fat trees [10], are suitable communication infrastructures for general-purpose single-chip computing platforms. They are more scalable for different system sizes and different performance requirements than fat trees, which has been achieved by allowing switches with different number of ports to be used in different switch stages of these hierarchical networks. Simulation results presented in this paper show that the XGFTs are able to produce good performance with random traffic patterns produced typically by randomly mapped processes. The results show also that the performance can be improved by mapping the communicating processes to the computing leaf nodes suitably in such a way that they produce more local cluster traffic.

XGFTs can be constructed from separate up-routing and down-routing networks with small switch blocks which are connected to each other with one or more unidirectional channels within switch nodes called dual-switch nodes in this paper. This arrangement has at least three advantages. Firstly, the small switch blocks of the dual-switch nodes are easier to design to achieve higher operation speed than larger switch nodes which contain only one large switch block and which are called mega-switches in this paper. Secondly, it is easier to place and route the small switch blocks than mega-switches and implement both dense and fast circuit layout. Thirdly, the up-routing and down-routing networks use different routing mechanisms which are simpler to implement separately with two distinct switches. One of the disadvantages of this arrangement is that commonly used shortest-path routing algorithms like Turnaround [11] do not produce good performance with it. In addition, the Turnaround and other similar earlier presented routing algorithms [12, 13, 14] can be used only in such BMINs where all of the switch nodes have the same number of ports, which limits their usability and makes them basically unusable in XGFTs. The number of ports must also usually be equal to 2^k for some positive integer k, so that these routing algorithms would be usable. However, these problems can be eliminated with an adaptive Turn-Back-When-Possible (TBWP). It can be shown that the performance of fat trees and XGFTs implemented with dual-switch nodes can be considerably improved by connecting the up-routing network to the down-routing network with extra links on top of the networks and by using the TBWP routing algorithm instead of shortest-path routing algorithms [9, 15].

In addition to the TBWP, this paper presents also a new adaptive shortest-path routing algorithm named Turn-Back (TB) which is usable in all of the XGFTs constructed from mega-switch nodes. One of the main objectives of this paper is to show that the TBWP and the TB have simple and feasible implementations for all kind of XGFTs independently of their topology. This is achieved by encoding the addresses of the communicating leaf nodes suitably in order to eliminate complex computations from the routing decision functions of the TBWP and the TB. The other objective of this paper is the evaluation and comparison of the performances of these two routing algorithms by simulations. The results of these simulations show that the new TB routing algorithm used with mega-switch nodes is able to produce higher performance than the TBWP used with dual-switch nodes.

This paper is organized as follows. Section 2 presents the topology of the XGFT and shows how the XGFTs can be generated top-down recursively. The same section presents also the architectures of the dual-switch and mega-switch nodes. Section 3 presents practical implementations of the TBWP and the TB routing algorithms which use encoded addresses. The recursive generation rules presented in Section 2 are used for explaining how the address space can be encoded. In addition, it is shown in Section 3 that the TBWP and TB route packets correctly to the destinations. This is done by showing that the original TBWP, which does all of the necessary computations by itself, routes packets correctly. Section 4 presents simulation results which are used for evaluating the performances of the routing algorithms. The performances are also compared in this same section. Finally, Section 5 concludes this paper.

2 THE TOPOLOGY OF THE XGFT

EXtended Generalized Fat Trees (XGFT) are Bidirectional Multistage Interconnection Networks (BMIN) which can be scaled for different systems sizes and different performance requirements. This has been achieved by allowing switches in different switch stages of the network to have different numbers of bi-directional ports. Owing to the regular topology, the XGFTs can be generated following certain rules which define the number of switch nodes in different switch stages of the network and how these switch nodes are connected to each other. The first of the following two subsections presents the topology of the XGFTs. It presents also the architecture and operation of the switch nodes. The second subsection presents generation rules which can be used for generating the XGFTs top-down recursively. The terms "stage" and "switch stage" are used equally in the following text.

2.1 The Topology and Structure of the XGFTs

The XGFTs are hierarchical BMINs where switches in different stages of the network can have different numbers of ports. The XGFT of height h can be defined as a tuple XGFT$(h, m_1, m_2 \ldots, m_h, w_1, w_2 \ldots, w_h)$ where switch nodes in stage i $(1 \leq i \leq h)$ have been connected to m_i child nodes and w_i parent nodes with bi-directional links. Switches in stage one, which is the lowest switch stage, have been connected to leaf nodes which use the network for communicating with each others. The leaf nodes can be connected to only one of the switches of stage one. The height h, which is the number of switch stages, and numbers m_i and w_i specify unambiguously the topology of the XGFT. Although this is a slightly different definition from the original definition of the XGFT [8], it still defines basically the same network topology.

Figure 1 a) depicts XGFT$(3, 4, 3, 5, 2, 2, 2)$ which has three stages of switches and 60 leaf nodes. Switches in different stages of the network have been numbered with the corresponding stage numbers from one to three. Small black squares below switch nodes are communicating leaf nodes. The overall number of leaf nodes can be expressed as the product $m_1 \times m_2 \times \ldots \times m_h$. For example, the number of leaves of the XGFT$(3, 4, 3, 5, 2, 2, 2)$ is $4 \times 3 \times 5 = 60$. It is assumed that the leaves are numbered from left to right in ascending order in such a way that the leftmost leaf has address zero and the rightmost leaf has address $m_1 \times m_2 \times \ldots \times m_h - 1$ like Figure 1 a) illustrates. Figure 1 b) on the right illustrates how the topology of the XGFTs can be changed. The XGFT$(3, 4, 3, 5, 3, 1, 2)$ in Figure 1 b) has still 60 leaf nodes, but the number of switches in stage two has been increased in order to increase the amount of routing resources for local traffic within the five sub-XGFT$(2, 4, 3, 2, 2)$s of height two. At the same time the number of topmost switches, which connect the sub-XGFTs of height two, has decreased from four to three. Generally speaking, the network on the right has more routing capacity for so called cluster traffic than the network on the left. This simple example illustrates how XGFTs can be suitably constructed for various applications which produce different network traffic.

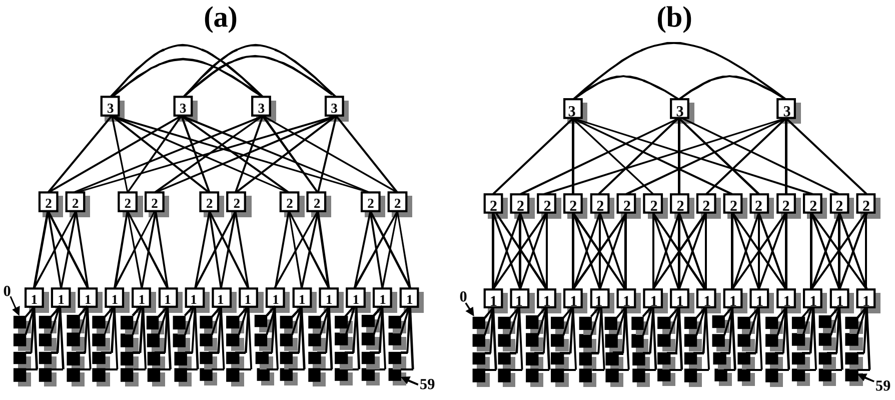

Fig. 1. a) XGFT$(3, 4, 3, 5, 2, 2, 2)$ and b) XGFT$(3, 4, 3, 5, 3, 1, 2)$

In XGFTs the switch nodes are connected to each other with bi-directional links. They have bi-directional ports for connecting to links which consist of two unidirectional channels for transferring data to opposite directions. This is illustrated in Figures 2 a) and 2 b) which depict a) dual-switch and b) mega-switch nodes. Switch nodes are connected to their parent nodes through bi-directional P-ports and to their child nodes through bi-directional C-ports. Each of the bi-directional P-ports consists of a pair of unidirectional input P_{DR}-port and output P_{UR}-port. Similarly bi-directional C-ports consist of unidirectional C_{UR}-ports and C_{DR}-ports. The numbering of the bi-directional P and C ports shows which parent and child nodes these ports are connected to. The index numbers are within value ranges 0 to $w_i - 1$ and 0 to $m_i - 1$ respectively $(1 \leq i \leq h)$. For example, switches send packets to parent number zero through $P_{UR}[0]$-port, and receive packets from it through $P_{DR}[0]$-port. They receive packets respectively from child node number three through $C_{UR}[3]$-ports, and send packets to it through $C_{DR}[3]$-port.

Switch nodes can be implemented in two different ways. Figure 2 a) depicts a dual-switch node which has been constructed from two separate switch blocks. The switch block on the left routes packets upwards in the network and the switch block on the right routes them downwards. These switch blocks can be called up-routing and down-routing switches, respectively. The up-routing switch is connected to the down-routing switch with one or more unidirectional channels named Turn-Back (TB) channels which are used for routing packets from the up-routing switch to the down-routing switch. Within the switch blocks the input port blocks (IP) are connected to the output port bocks (OP) through crossbars (CB). In the topmost switch stage the number of connections between the up-routing and down-routing switches can be increased easily by connecting free P_{UR}-ports to free P_{DR}-ports. Otherwise, these ports would remain unconnected and unused, because switches in the topmost stage do not have parent nodes. Although in Figures 1 a) and 1 b) these connections have been drawn with extra links between different topmost switch nodes of the networks, it would also be possible to connect the P_{UR}-ports and the P_{DR}-ports of the same topmost switch nodes. Figure 2 b) depicts a mega-switch node where input ports are connected directly to all of the appropriate output ports over one crossbar (CB) within one switch block. Owing to these direct connections from input ports to output ports the extra links would be useless on top of the networks constructed from mega-switches. For this reason, the topmost root switches of such networks can be implemented without the P_{UR} and P_{DR}-ports which are used for connecting to parent nodes. Because XGFTs constructed from different switch nodes are actually two different networks, they need also different routing algorithms like TBWP or TB in order to work properly and produce good performances.

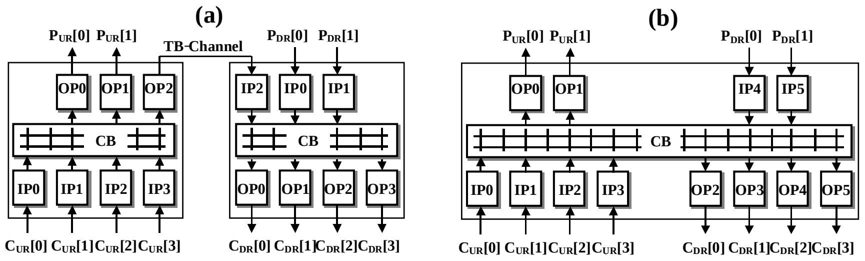

Fig. 2. The architectures of the a) dual-switch and b) mega-switch nodes

2.2 Recursive Generation of the XGFTs

Figures 1 a) and 1 b) depict simple XGFT networks which can be generated top-down recursively. Network depicted in Figure 1 a) was also used in simulations which were used for evaluating the performances of the TBWP and the TB routing algorithms. The recursive generation of the XGFTs starts from the top of the XGFT$(h, m_1, m_2, \ldots, m_h, w_1, w_2, \ldots, w_h)$ of height h, and continues in the similar way within each of its sub-XGFT$(L, m_1, m_2, \ldots, m_L, w_1, w_2, \ldots, w_L)$s of height L $(1 \leq L \leq h-1)$ until finally within sub-XGFT$(1, m_1, w_1)$s of height one only one switch is generated. The generation proceeds stage by stage through the whole XGFT. The topmost root switches of the XGFTs of height h are generated at first. In the next step sub-XGFTs of height $h-1$ are generated and connected to the root switches. After this the generation continues recursively within each of the sub-XGFTs of height $h-1$.

Connections between the root switch nodes in stage h and sub-XGFTs of height $h-1$ follow certain rules. Firstly, each of the root switch nodes is connected to all of the sub-XGFTs, and secondly, each of the sub-XGFTs is connected to all of the root switch nodes. For example, there are four topmost root switches in stage three, and five sub-XGFT$(2, 4, 3, 2, 2)$s of height two in the XGFT$(3, 4, 3, 5, 2, 2, 2)$ depicted in Figure 1 a). All of the root switches have been connected to all of the sub-XGFTs, and all of the sub-XGFTs have been connected to all of the root switch nodes, like Figure 1 a) illustrates. Below is a list of various properties of the XGFT and the sub-XGFTs of height $L+1$ $(1 \leq L \leq h-1)$ which are used in defining recursively the structure of the XGFTs. This list shows that the root switches and the sub-XGFTs have appropriate number of ports for connecting to each other. In the following it is also assumed that each pair of unidirectional P_{UR} and P_{DR}-ports forms one bidirectional P-port, and that each pair of unidirectional C_{UR} and C_{DR}-ports forms one bidirectional C-port.

1. The number of root switches of the sub-XGFTs of height $L + 1$: $R_{L+1} = w_1 \times w_2 \times \ldots \times w_{L-1} \times w_L$.
2. The number of C_{UR} and C_{DR}-ports of the root switches in stage $L + 1$: $C_{L+1} = w_1 \times w_2 \times \ldots \times w_{L-1} \times w_L \times m_{L+1}$.
3. The number of sub-XGFTs of height L in sub-XGFT of height $L+1$: $X_L = m_{L+1}$.
4. The number of P_{UR} and P_{DR}-ports of the root switches of the sub-XGFTs of height L: $P_L = w_1 \times w_2 \times \ldots \times w_{L-1} \times w_L = R_L \times w_L$.

For example, in Figure 1 a) $R_3 = 2 \times 2 = 4$ which is the number of topmost root switch nodes, and $X_2 = m_3 = 5$, which is the number of sub-XGFTs of height two. The overall number of P_{UR} and P_{DR}-ports $(P_L \times X_L)$ of all of the sub-XGFTs of height L is equal to the overall number of C_{UR} and C_{DR}-ports (C_{L+1}) of all of the root switches within each of the sub-XGFTs of height $L + 1$. In addition, the number of top-most root switches (R_{L+1}) of the sub-XGFTs of height $L + 1$ is equal to the number of P_{UR} and P_{DR}-ports (P_L) of each of the sub-XGFTs of height L. Furthermore, X_L is equal to m_{L+1}. The following two simple rules define how the connections between the top-most root switches and the sub-XGFTs of height L are generated within each of the sub-XGFTs of height $L + 1$.

1. Up-routing port $P_{UR}[j][i]$ $(0 \leq i \leq R_{L+1}-1)$ of sub-XGFT j $(0 \leq j \leq m_{L+1}-1)$ of height L is connected to port $C_{UR}[i][j]$ $(0 \leq j \leq m_{L+1} - 1)$ of switch i $(0 \leq i \leq R_{L+1} - 1)$ in stage $L + 1$.
2. Down-routing port $C_{DR}[i][j]$ $(0 \leq j \leq m_{L+1} - 1)$ of the root switch i $(0 \leq i \leq R_{L+1} - 1)$ in stage $L + 1$ is connected to port $P_{DR}[j][i]$ $(0 \leq i \leq R_{L+1} - 1)$ of sub-XGFT j $(0 \leq j \leq m_{L+1} - 1)$ of height L.

Fat trees and XGFTs can be generated top-down recursively according to the above properties and rules with hardware description languages like VHDL which can be used for generating recursive structures [16]. Figure 3 shows a piece of pseudo code which generates up-routing $(C_{UR} \Leftarrow P_{UR})$ and down-routing $(P_{DR} \Leftarrow C_{DR})$ channels between the root switch nodes and the sub-XGFTs of height L within

XGFTs of height $L + 1$. This piece of generator code is executed repeatedly within each of the sub-XGFTs when the XGFTs are generated top-down recursively. The bi-directional P-ports of the top-most root switches in stage $L + 1$ are connected to bi-directional P-ports of the sub-XGFT of height $L + 1$. For example, port $P[i][j]$ $(0 \leq i \leq w_1 \times w_2 \ldots \times w_L - 1,\ 0 \leq j \leq w_{L+1} - 1)$ of root switch i in stage $L + 1$ is connected to port $P[k]$ $(k = i \times w_{L+1} + j)$ of the sub-XGFT of height $L + 1$. The bi-directional C-ports of the switches in stage one of the sub-XGFTs are connected to the bi-directional C-ports of the XGFT within appropriate index ranges. If the height of the sub-XGFT is L $(1 \leq L \leq h - 1)$ and its ordinal number is p $(0 \leq p \leq m_h \times m_{h-1} \times \ldots \times m_{L+1} - 1)$, this index range is $p \times m_L \times m_{L-1} \times \ldots \times m_1$ to $(p + 1) \times m_L \times m_{L-1} \times \ldots \times m_1 - 1$. The leaf nodes are connected to the XGFT through these C-ports.

```
Connect switch i : FOR i IN 0 TO $R_{L+1}$ −1 GENERATE
        Connect sub-XGFT j : FOR j IN 0 TO $m_{L+1}$ −1 GENERATE
                "Port $C_{UR}$[i][j] of switch i in stage L+1 <= Port $P_{UR}$[j][i] of sub-XGFT j of height L";
                "Port $P_{DR}$[j][i] of sub-XGFT j of height L <= Port $C_{DR}$[i][j] of switch i in stage L+1";
        END GENERATE Connect sub-XGFT j ;
END GENERATE Connect switch i ;
```

Fig. 3. A code segment which generates links between the root switches and sub-XGFTs within XGFTs

3 THE OPERATION OF THE TBWP AND THE TB ROUTING ALGORITHMS

The TBWP was designed for improving the performance of such XGFTs which have been constructed from dual-switch nodes where shortest-path routing algorithms like the TB and Turnaround [11] do not work properly and produce good performance. The TB routing algorithm, which was designed for XGFTs constructed from mega-switches, is almost similar to the TBWP. The routing decision functions of both of these routing algorithms would use integer division and multiply operations for computing the routing decisions if their implementations would not have been simplified with address space encoding. This would have been a problem, because the integer multiply and division operations do not have feasible hardware realization usable in small switches of the on-chip networks. Owing to the address space encoding, the routing decison functions of both of the routing algorithms can be implemented with simple comparison and cut operations. The first of the following two subsections describes the simplified versions of the TBWP and TB routing algorithms which operate with encoded addresses, and the next subsection presents the original TBWP routing algorithm which operates with unencoded integer addresses. These subsections also show how the address space is encoded and why the TBWP and TB route packets correctly.

3.1 TBWP and TB Routing Algorithms with Encoded Address Spaces

There is exactly one routing path of length h from each of the top-most root switches to each of the leaf nodes D $(0 \leq D \leq m_1 \times m_2 \times \ldots \times m_h - 1)$ in the XGFTs of height h. This path can be unambiguously defined with output port numbers d_L $(1 \leq L \leq h)$ of the switches along the path starting from any of the top-most root switches. This follows from the rules presented in Section 2 which define how XGFTs are generated top-down recursively. According to these rules output ports $C_{DR}[j]$ $(0 \leq j \leq m_{L+1} - 1)$ of all of the root switches of the XGFT and the sub-XGFTs of height $L+1$ $(1 \leq L \leq h-1)$ are connected to sub-XGFT j $(0 \leq j \leq m_{L+1} - 1)$ of height L. For this reason, the routing path from any of the topmost roots of the XGFT to the destination leaf node can be represented as an output port number sequence $d_h d_{h-1} \ldots d_1$ where numbers d_L $(1 \leq L \leq h)$ are output C_{DR}-port numbers of the switches in stages L $(1 \leq L \leq h)$. The width of the binary number representation of the numbers d_L depends on the number of C_{DR}-ports (m_L) of the switch nodes in stage L. It is equal to the smallest integer k_L for which m_L is smaller than or equal to two to the power of k_L, i.e. the smallest integer k_L for which $m_L \leq 2^{k_L}$. For this reason, addresses D $(0 \leq D \leq m_1 \times m_2 \times \ldots \times m_h - 1)$ can be unambiguously encoded by computing the k_L bits wide output C_{DR}-port numbers d_L and by concatenating them. In other words, the encoded form D_{enc} of address D is an output port number sequence $d_h d_{h-1} \ldots d_1$ where the width of numbers d_L $(1 \leq L \leq h)$ is equal to k_L which is the smallest such an integer for which m_L is smaller than or equal to two to the power k_L. For instance, if switches in stage L have five $(m_L = 5)$ C_{DR}-ports, the binary number representation of number d_L requires always three $(k_L = 3)$ bits, because three is the smallest integer value of parameter k_L for which $m_L \leq 2^{k_L}$ $(5 < 8 = 2^3)$.

Below is a simplified version of the TBWP which operates with encoded source S and destinations D addresses $S_{enc} = s_h s_{h-1} \ldots s_1$ and $D_{enc} = d_h d_{h-1} \ldots d_1$ which the packets carry in their headers when they are routed across the network. This description presents the operation of the routing decision function of the switches in the switch stage L of the XGFT. Part 1 of the TBWP is used in the input port blocks connected to input C_{UR}-ports of the dual-switch node depicted in Figure 2 a), and Part 2 of the TBWP is used in the input port blocks connected to input P_{DR}-ports.

TBWP routing algorithm with encoded addresses:

Part 1 (Up-routing): If a packet arrives at the up-routing switch block of the dual-switch node in stage L $(1 \leq L \leq h-1)$ through any of the C_{UR}-ports and $s_h s_{h-1} \ldots s_{L+1} = d_h d_{h-1} \ldots d_{L+1}$, route it to the turn-back channel. If the turn-back channel is already reserved, then route the packet to one of the parent nodes through any of the P_{UR}-ports. If sequence $s_h s_{h-1} \ldots s_{L+1}$ is not equal to $d_h d_{h-1} \ldots d_{L+1}$, route the packet to one of the parent nodes through any of the P_{UR}-ports. In the top-most stage (L = h) route packets always either through the turn-back channel to the down-routing switch or through any of the P_{UR}-ports.

Part 2 (Down-routing): If a packet arrives at the down-routing switch of the dual-switch node in stage L $(1 \leq L \leq h)$ through any of the P_{DR}-ports or through the turn-back channel, route it downwards through $C_{DR}[d_L]$-port.

Part 1 of the TBWP algorithm above routes packets upwards in the XGFTs. It implements an adaptive routing, because it is able to adapt its routing decisions to the state of the switch nodes. It depends on the operation of the arbiters how the transfers are scheduled from input ports to free P_{UR}-ports and turn-back-channels. If the switch node is one of the common ancestors of both the source and destination leaf nodes, Part 1 routes the packet downwards when it is possible. The switch is one of the common ancestors, if the most significant numbers of the encoded source and destination addresses are equal, i.e. if $s_h s_{h-1} \ldots s_{L+1} = d_h d_{h-1} \ldots d_{L+1}$. This is because the sequences $s_h s_{h-1} \ldots s_{L+1}$ and $d_h d_{h-1} \ldots d_{L+1}$ determine unambiquously the sub-XGFTs of height L where the source and destination nodes are. Therefore, the routing decisions can be based on testing the equality of the most significant $k_h + k_{h-1} + \ldots + k_{L+1}$ bits of the source and destination addresses. Part 2 of the TBWP simply cuts the output port numbers from the destination address. It routes packets downwards deterministically, because output port numbers determine completely the routing path downwards.

The TBWP can be easily modified to the new TB shortest-path routing algorithm the operation of which is described below. The adaptive TB was designed for the XGFTs constructed from mega-switch nodes depicted in Figure 2 b).

TB routing algorithm with encoded addresses:

Part 1 (Up-routing): If a packet arrives at a mega-switch node in stage L $(1 \leq L \leq h-1)$ through any of the C_{UR}-ports and $s_h s_{h-1} \ldots s_{L+1} = d_h d_{h-1} \ldots d_{L+1}$, then route it downwards through $C_{DR}[d_L]$-port. Otherwise, route the packet upwards through any of the P_{UR}-ports. In the top-most stage $(L = h)$ route packets always downwards through $C_{DR}[d_h]$-port.

Part 2 (Down-routing): If a packet arrives at a mega-switch node in stage L $(1 \leq L \leq h-1)$ through any of the P_{DR}-ports, then route it downwards through $C_{DR}[d_L]$-port.

3.2 Deadlock-Free Routing

Like Turnaround, both TBWP and TB are deadlock-free routing algorithms. This is because, like in all of the networks wich have a treelike acyclic topology, in the XGFTs none of the routing paths can start and end at the same network node or contain cycles. In addition, the operation of the TBWP and the TB consists of distinct up-routing and down-routing phases, and they switch the packets from the up-routing paths to the down-routing paths only once when they route them from sources to destinations. For this reason, there can be no cycles in the dependency graphs of the possible routing paths either, and it is impossible that packets would wait in a cycle for each other to be routed forward. As a consequence of this, the TBWP and the TB must be deadlock-free routing algorithms [17]. In addition, because the TB turns packets always downwards from the nearest common ancestors of the source and destination nodes, it must be a shortest-path routing algorithm. This is because the TB turns packets back from the first switch node where condition $s_h s_{h-1} \dots s_{L+1} = d_h d_{h-1} \dots d_{L+1}$ is true. Because C_{DR}-port number sequences $s_h s_{h-1} \dots s_{L+1}$ and $d_h d_{h-1} \dots d_{L+1}$ determine unambiguously the sub-XGFT of height L the leaves of which both source and destination leaf nodes are, this switch node is one of the roots of sub-XGFT of height L and it must be one of the nearest common ancestors too.

3.3 Operation Examples with Encoded Addresses

Figure 4 illustrates the encoding of address 11 in XGFT$(3, 4, 3, 5, 2, 2, 2)$ where down-routing paths from the topmost roots to leaf node 11 are drawn with dot lines. It is assumed that all of the ports of the switch nodes have been numbered in an ascending order starting from the left like Figure 2 depicts. As one can see, all of the top-most roots are connected through $C_{DR}[0]$-port to the same sub-XGFT of height two the leaf of which leaf number 11 is. Switches in stage two along all of the down-routing paths are connected through $C_{DR}[2]$-port to the last switch in stage one which is connected to the destination leaf node through $C_{DR}[3]$-port. Thus, the encoded address is output port number sequence $d_3 d_2 d_1 = (0, 2, 3)_{enc}$. The encoded address is represented here as a three tuple where the individual output port numbers have been separated with commas from each other, and the number sequence has been enclosed to brackets in order to separate it from the other parts of the text. The precomputed encoded addresses can be stored into read-only-memories (ROM) placed at packet sources which can use them for fast address translations without complex computations.

Figure 4 illustrates also an example of how the TBWP can route a packet from leaf processor node 27 to processor node 35 in XGFT$(3, 4, 3, 5, 2, 2, 2)$. In this example it is assumed that the network has been constructed from dual-switch nodes where up-routing and down-routing switches have been connected with only one turn-back channel. The encoded addresses of the leaf nodes 27 and 35 are tuples $(2, 0, 3)_{enc}$, and $(2, 2, 3)_{enc}$ in a respective order. The routing path from the source to destination has been drawn with dashed line arrows, and switches along the routing path have been numbered with numbers one to six. Switch number one along the routing path routes the packet upwards, because most significant numbers of the encoded addresses are not equal, i.e. $(2, 0)_{enc} \neq (2, 2)_{enc}$. This is because the source and destination nodes are not leaves of the same sub-XGFT of height one. Although $(2)_{enc} = (2)_{enc}$, which indicates that the source and destination nodes are leaves of the same sub-XGFT of height two, switch number two routes the packet also upwards. This is because the turn-back channel of the dual-switch node is al-

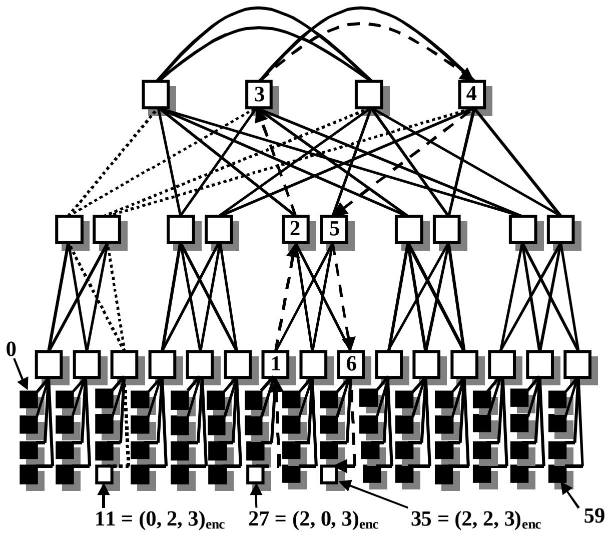

Fig. 4. Examples of the address space encoding and the operation of the TBWP

ready reserved and it can not be used for routing the packet to the down-routing switch. Switch number three routes the packet upwards to another root switch, because of this same reason. Switch number four routes the packet through output port $C_{DR}[2]$ to switch number five which routes it to switch number six through output port $C_{DR}[2]$. Finally, switch number six routes packet to its destination through output port $C_{DR}[3]$. The shortest-path TB would route the packet downwards from switch number 2 which is the nearest common ancestor of the source and destination leaf nodes. Furthermore, it would not be able to use additional links on top of the networks if the packet would be routed via any of the topmost root switches.

3.4 The Original TBWP Routing and Address Space Encoding

The operation of the TBWP and TB routing algorithms presented above is based of the operation of the original TBWP which uses unencoded integer addresses and which was designed for XGFTs constructed from dual-switch nodes. The routing decisions of the original TBWP are based on computations which can also be used for address space encoding. The leaf addresses can be encoded with Part 2 of the original TBWP routing algorithm in the following way in three steps. In the first step the output C_{DR}-port numbers d_L are computed for all L $(1 \leq L \leq h)$ with a formula $d_L = (D\,DIV\,m_0 \times m_1 \times m_2 \times \ldots \times m_{L-1})\,MOD\,m_L$ where DIV- and MOD-operations compute integer quotient and remainder of the division of two integers. In the second step numbers d_L are converted to binary numbers the width of which is the smallest positive integer k_L for which $m_L \leq 2^{k_L}$. In the third step, the encoded binary number representations of the output port numbers d_L are concatenated to form the encoded address $D_{enc} = d_h d_{h-1} \dots d_1$. In the description of the original TBWP below parameters S and D are unencoded source and destination addresses carried in the packet headers, and parameter $m_0(= 1)$ has been inserted to formulas so that Part 2 would work properly also in stage one.

Original TBWP routing algorithm:

Part 1 (Up-routing): If a packet arrives at the up-routing switch block of the dual-switch node in stage L $(1 \leq L \leq h)$ through one of the C_{UR}-ports and $S DIV m_0 \times m_1 \times m_2 \times \dots \times m_L = D\, DIV\, m_0 \times m_1 \times m_2 \times \dots \times m_L$, then route the packet through the turn-back channel to the down-routing switch block, else route it through one of the P_{UR}-ports. If the turn-back channel is already reserved, then route the packet through any of the P_{UR}-ports.

Part 2 (Down-routing): If a packet arrives at the down-routing switch block of the dual-switch node in stage L $(1 \leq L \leq h)$ through one of the P_{DR}-ports or through the turn-back channel, then route it through $C_{DR}[(D\, DIV\, m_0 \times m_1 \times m_2 \times \dots \times m_{L-1})\, MOD\, m_L]$-port downwards.

It can be shown that the presented simplified versions of the TBWP and TB algorithms route packets correctly by showing that the original TBWP routes packets to their desired destinations in the XGFTs and especially that its Part 2 computes output C_{DR}-port numbers correctly.

Assume that the leaf nodes have been given addresses in ascending order starting from the left in such a way that the leftmost leaf has the smallest address zero and the rightmost leaf has he highest address $m_1 \times m_2 \times \dots \times m_h - 1$ like Figure 1 illustrates. Let $N_{L-1} = m_0 \times m_1 \times \dots \times m_{L-1}$ be the number of leaves in each of the sub-XGFTs of height $L - 1$ the root of which the node in stage L is. Then $N_L = N_{L-1} \times m_L = m_0 \times m_1 \times \dots \times m_{L-1} \times m_L$ is the total number of leaves in all of the sub-XGFTs connected to the switch node in stage L. In Part 1 condition $S\, DIV\, N_L = S\, DIV\, m_0 \times m_1 \times \dots \times m_L = D\, DIV\, m_0 \times m_1 \times \dots \times m_L = D\, DIV\, N_L$ tests whether the source and destination nodes are both leaves of the sub-XGFTs connected to the switch node in stage L. In order to show this assume that the destination address $D = p_D \times N_L + q_D$ and the source address $S = p_S \times N_L + q_S$ where numbers p_D and p_S $(0 \leq p_D, p_S \leq m_{L+1} \times m_{L+2} \times \dots \times m_h - 1)$ are ordinal numbers of the sub-XGFTs of height L, and numbers q_D and q_S $(0 \leq q_D, q_S \leq N_L - 1)$ are ordinal numbers of the source and destination leaves of the sub-XGFTs of height L. Then $D\ DIV\ N_L = p_D = p_S = S\ DIV\ N_L$ only if destination and source nodes are leaves of the same sub-XGFT of height L. For this reason, both of them must be leaves of the sub-XGFTs of height $L-1$ connected to the same switch node, and the packet can be switched to the turn-back channel if the condition is true. Otherwise it must always be sent to one of the parent switches so as to find the common ancestor switch node. In Part 2 the destination address D can also be written in the form $D = p \times N_{L-1} + q$, where q and p are such integers that $0 \leq q \leq N_{L-1} - 1$ and $0 \leq p \leq m_L \times m_{L+1} \times \dots \times m_h - 1$ where $m_L \times m_{L+1} \times \dots \times m_h$ is the total number of sub-XGFTs of height $L-1$. Number p is the ordinal number of the sub-XGFT of height $L-1$ the leaf of which the destination leaf node is. Thus, if the destination address D is divided by N_{L-1}, then the quotient $D\,DIV\,N_{L-1} = p$ is the ordinal number of the destination sub-XGFT of height $L-1$, and the remainder $p\,MOD\,m_L$ is the appropriate output C_{DR}-port number of the down-routing switch in stage L.

Figure 5 illustrates how the TBWP computes the output port numbers in Part 2. In this example source (S) and destination (D) leaf nodes are in adjacent sub-XGFTs of height $L-1$. Since the leaves have been numbered in ascending order starting from the left, the source address S is smaller number than the destination address D. The ordinal numbers p_S and p_D of the sub-XGFTs can be computed by dividing the addresses by the number of leaf nodes of the sub-XGFTs of height $L-1$, i.e. by N_{L-1}. This is what the TBWP does in Part 2 to destination address D with $D\,DIV\,m_0 \times m_1 \times \ldots \times m_{L-1} = p_D$. The output port numbers are remainders s_L and d_L which can be computed by dividing the ordinal numbers by m_L which is the number of sub-XGFTs of height $L-1$ in each of the sub-XGFTs of height L. The TBWP does this with formula $d_L = p_D\,MOD\,m_L = (D\,DIV\,m_0 \times m_1 \times m_2 \times \ldots \times m_{L-1})\,MOD\,m_L$. All of the switches in stage L do the same computations, although in Figure 5 it is assumed that a packet is switched through switch zero from sub-XGFT p_S to sub-XGFT p_D.

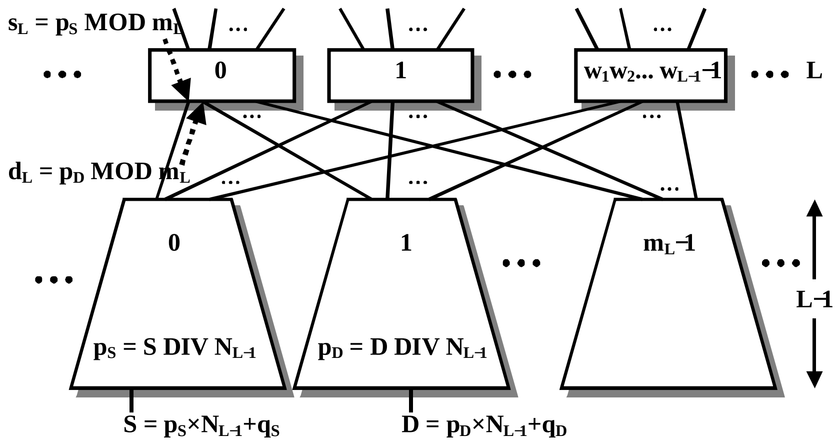

Fig. 5. Routing in sub-XGFT of height L

4 PERFORMANCE EVALUATION

The performances of the TBWP and the TB routing algorithms were evaluated by simulating their operation with different network configurations. Simulations were performed with high-abstraction level behavioral models so as to shorten the simulation time, and because more accurate Register-Transfer-Level (RTL) models would not have produced any useful additional information essential for the comparison of the routing algorithms. Because the behavioral models are faster to simulate than RTL models, they could also be used for speeding up simulations of complete systems where part of the components could also be RTL models.

These performance simulations were accomplished with two different traffic patterns. The first simulations with uniformly distributed artificial traffic can be considered a kind of general case where randomly placed processes send packet traffic to each other. Simulations with more local cluster-traffic can be considered a more realistic case. This is because in true systems communicating hardware blocks and software processes executed by processors are often placed as near to each other as possible in order to prevent traffic from spreading all over the network, which reduces the amount of required hardware resources and communication latencies.

4.1 Implementation of Simulations

Simulations were done with XGFT$(3, 4, 3, 5, 2, 2, 2)$ depicted in Figure 1 a) with dual-switch and mega-switch nodes. Although in Figures 1 a) and 1 b) additional bi-directional links on top of the network connect different topmost root switches to each other, simulations were done with such networks where P_{UR}-ports and P_{DR}-ports of the same topmost root switches were connected to each other with uni-directional channels. Network constructed from dual-switches was simulated with both the TBWP and the TB routing algorithms. For this reason, the TB was also modified for dual-switches so that it was possible to compare the performance of the TBWP with that of some shortest-path routing algorithm. Network constructed from mega-switches was simulated with the TB routing algorithm presented in subsection 3.1 only. Simulation time was divided into time slots which were equally long with operation cycles (or clock cycles) of the switch nodes.

Switch nodes were modelled as input-output buffered crossbar switches with input and output buffers of only eight words, which was also the size of the smallest packets used in simulations. It was possible to use buffers which were smaller than the average packet size, because switches used wormhole routing strategy. Wormhole switches do routing decisions according to the routing information carried in the packet headers immediately after they have received the headers and route packets forward. It would have been possible to use larger buffers in the switch nodes in order to improve the performances. However, larger buffers are practically unusable, because the increment of the buffer size would also increase the size of the real NOC.

In packet switched networks packets are usually routed through switches in three steps. In the first step the packets arrive at the input ports of the switch, and routing algorithm does routing decisions according to the information carried in the packet headers. In the second step the switch arbiter schedules appropriate output ports so that the packets can be sent to the next switch, and in the third step the packets are transferred from the input ports to the output ports. The behavioral models of the switches operated slightly differently. In the behavioral switch models the same processes did the routing decisions and scheduled several packet transfers from the input ports to output ports during one time slot. This process used rotating priority (round-robin) arbitration to choose the next input and output ports for which it scheduled the transfers. After this another process performed the transfers from input ports to output ports.

Packets were transferred from one switch to another across asynchronous channels. Instead of transferring packets word by word over the channels a record type data-signal was used for this purpose. These record type packets contained distinct fields for source and destination addresses, packet length, and data. The transmitting side started the transfer by putting the packet on the data-signal. The transfer of distinct words was modeled with request-signal pulses the number of which was equal to the length of the packet. The length of the request-signal pulse was half of the length of the time slot, and the transfer of one word took time one time slot. The receiving side could stop the transfer by setting a ready-signal to low, and it allowed the transfer to be continued again by setting the ready-signal to high. Both the transmitting and the receiving sides of the channels used counters for counting the number of pulses and for controlling the starting and ending of the transfer of the packets.

Simulations were performed with artificially generated packet traffic. There were separate traffic sources for generating and transmitting traffic to each of the input ports of the XGFTs in the simulated system. Traffic sources generated new packets only when they were either transmitting or idle, i.e. during the time slots when the ready-signal was high and it would have been possible to transmit packets to the network. During these time slots they generated a uniformly distributed pseudo random number prn which got values from real number interval $[0.0, 1.0]$. This pseudo random number prn was compared to a quotient $\rho/averpcktlen$ where parameter ρ was a load factor and parameter $averpcktlen$ was the average packet length which was 20 words in these simulations. Traffic sources generated a new packet and stored it into a packet queue, if prn was smaller than or equal to the quotient $\rho/averpcktlen$. Another uniformly distributed pseudo random number, which got values 8 to 32, was used for generating the length of packets. Packet sources transmitted the next packet to the network immediately after the previous one if their packet queue was not empty.

This kind of traffic generation method implements also a feedback mechanism which affects sligthly the statistical properties of the generated traffic. However, because the same method was used in all of the simulations, its usage does not affect the order of superiority of different network configurations and routing algorithms, and in any case, the traffic was artificially generated for the purpose of the comparison of the performances only. In addition, it was possible to do the simulations without any knowledge of the performances and saturation points of the different network configurations by changing only ρ's value within the real number range 0.0 to 1.0. During simulations the utilization of the time slots usable for transmissions corresponded quite accurately to the value of parameter ρ when traffic sources loaded the network equally according to ρ's value. As ρ's value was increased the effective throughput also increased respectively. Near the point where the network became saturated and average routing delays grew very quickly the increment of ρ's value had only a negligible effect on the effective throughput. In this point the proportion of the transmission time to the overall time usable for transmissions was only slightly smaller than $\rho = 1.0$, and traffic sources transmitted practically as much traffic to the network as it was able to receive.

4.2 Simulation Results

In simulation results throughputs show the proportion of the average number of time slots which different traffic sources were able to use for transmission to the total number of time slots in percentages. Average latencies show the average number of time slots between the time instants of the transmission and reception of the first word of the packets. Simulations with all of the network configurations produced almost equal latencies before the point where the networks became saturated and latencies increased very quickly like e.g. in Figures 6 a) and 6 b) which show the results of the first simulations. These first simulations were performed with uniformly distributed traffic load, and their length was 250 000 time slots like the lenghts of all of the other simulations too. For example, an average throughput of 23.1 % per traffic source corresponds to an average transmission time of approximately 57 750 time slots per traffic source. This corresponds to a total of about 173 250 packets through the whole network when there are 60 traffic sources in the system and the average packet length is 20 words. This can be considered high enough number of packets for calculating reliable estimates of the average latencies and throughputs.

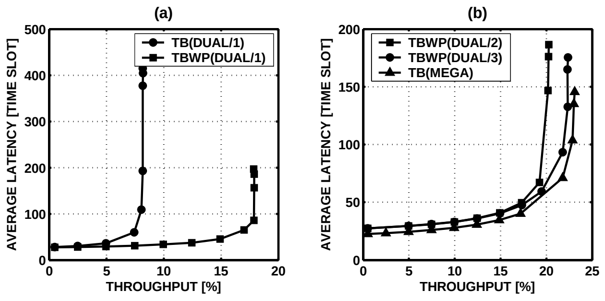

Fig. 6. Performances produced by the TB and TBWP routing in networks constructed a) from dual-switch nodes, and b) from mega-switches and dual-switches with different number of turn-back channels

Like Figure 6 a) and results in Table 1 show the TBWP (TBWP(DUAL/1)) produces twice as high maximum average throughput as the TB (TB(DUAL/1)) in a network constructed from such dual-switch nodes where the switch halves are connected with only one turn-back channel (TBC). This is because the TBWP can route packets through upper switch stages, if the turn-back channel of the nearest common ancestor is already reserved. The TBWP is also able to use additional channels within the top-most root switches, which the TB is not able to do. Figure 6 b) shows how this situation changes when the TB is used with mega-switches (TB(MEGA)). Like Figure 6 b) and results in Table 1 show it produces clearly higher maximum average throughput per source (TB(MEGA)) than the TBWP with dual-switch nodes. This is owing to the larger number of internal switching resources of the mega-switch nodes. However, the performance of the TBWP can be improved by increasing the number of turn-back channels between switch halves within dual-switch nodes from one to two (TBWP(DUAL/2)) or three (TBWP(DUAL/3)). With three turn-back channels (TBWP(DUAL/3)) the TBWP is able to produce almost equally high maximum average throughput as the TB (TB(MEGA)). The average latencies are not very much higher than those produced by the TB, although the TBWP must use also upper switch stages for routing the packets through the network.

Switch type and the number of TB-channels (TBC)	**Routing Algorithm**	**Max. Average Throughput [%]**	**Max. Average Latency [time slots]**
Dual with 1 TBC (DUAL/1)	TB	8.18	415.0
Dual with 1 TBC (DUAL/1)	TBWP	17.8	196.7
Dual with 2 TBCs (DUAL/2)	TBWP	20.2	186.6
Dual with 3 TBCs (DUAL/3)	TBWP	22.3	175.4
Mega with no TBCs (MEGA)	TB	23.1	145.6

Table 1. Simulated performances with uniformly distributed traffic load

Processes, which require a lot of bandwidth for communication with each other, will most probably be placed at neighboring processing nodes as near to each other as possible in order to reduce communication latencies and the amount of required network resources. In the systems where XGFTs are used the distance between communicating processes can be minimized by placing the processes at the leaves of the same sub-XGFTs. As a consequence of this, larger proportion of the traffic is transferred within the sub-XGFTs. Because there are five sub-XGFT$(2, 4, 3, 2, 2)$s with 12 leaf nodes in the XGFT$(3, 4, 3, 5, 2, 2, 2)$, the effect of the increased locality of the traffic on the performance was studied with simulations with clusters of 12 leaf processing nodes. Each of these five clusters was placed as a whole at the leaves of the same sub-XGFT$(2, 4, 3, 2, 2)$s.

In these simulations the TBWP was used with dual-switch nodes with one turn back channel, and the TB was used with mega-switches. These simulations were performed with a cluster load of 0.75, which means that in 75 % of the transmitted traffic was transferred within the clusters and only 25 % of it between the clusters. The simulated system produces clearly more local traffic with this cluster load value, because, for example, smaller cluster load of 0.2 would correspond to fully uniformly distributed traffic.

Like simulation results in Figure 7 a) show both the TBWP and TB produce higher throughputs as the traffic becomes more local. In addition, the TB produces clearly higher maximum average throughput of 40.7 % (TB) per traffic source than

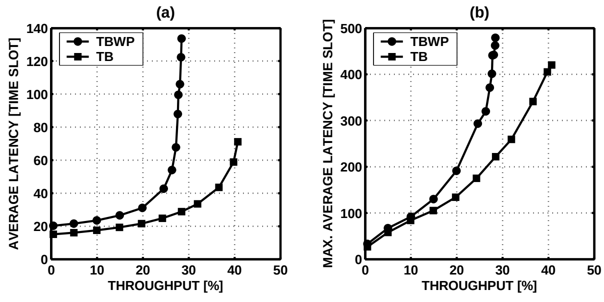

Fig. 7. a) Average delays and b) average maximum delays with cluster-traffic

the TBWP which produces maximum average throughput of 28.4 % (TBWP). Despite this, the TBWP does not produce considerably higher average routing delays than the TB before network becomes saturated. The corresponding average maximum routing delays, which are averages of 60 maximum delay values measured by the leaf computing nodes, are shown in Figure 7 b). As lines in Figure 7 b) show, the TBWP produces higher average maximum routing delays than the TB. This is because it must route packets via upper switch stages when the turn back channels are reserved. The difference between average routing delays (Figure 7 a)) and average maximum routing delays (Figure 7 b)) grows fastest between the points of 20 % and 28.4 % where the network constructed from dual-switch nodes saturated. As a summary it can be said that these results show that the TB is able to produce higher performances than the TBWP with both uniformly distributed and cluster-traffic, which is owing to the higher switching capacity of the mega-switches.

5 CONCLUSIONS

This paper presents simple and feasible implementations of the TBWP and the TB routing algorithms usable in the XGFTs constructed from dual-switch and mega-switch nodes. This is achieved by address space encoding which eliminates complex computations from the routing decision functions. Owing to the address space encoding routing decision functions can be implemented with simple compare and cut operations which have small and fast hardware implementations. According to the results of the performance simulations with behavioral models of different network configurations and with different traffic patterns the new TB, which is used in XGFTs constructed from mega-switch nodes, is able to produce higher performance than the TBWP, which is used in XGFTs constructed from dual-switches. Although this became evident especially with cluster-traffic, performances were still comparable. In addition, it would be possible to improve the performance of the TBWP by increasing the number of turn-back channels between the switch halves of the dual-switch nodes.

REFERENCES

[1] Zhang, H.—Prabhu, V.—George, V.—Wan, M.—Benes, M.—Abnous, A.—Rabaey, J.: A 1-V Heterogeneous Reconfigurable DSP IC for Wireless Baseband Digital Signal Processing. IEEE Journal of Solid State Circuits, Vol. 35, 2000, No. 11, pp. 1697–1704.

[2] Kumar, S.—Jantsch, A.—Soininen, J.-P.—Forsell, M.—Millberg, M.—Oberg, J.—Tiensyrja, K.—Hemani, A.: A Network on Chip Architecture and Design Methodology. Proceedings of Annual Symposium on VLSI, Pittsburgh, USA, 2002, pp. 105–112.

[3] Rabaey, J.: Chapter 1. System-on-Chip-Challenges in the Deep-Sub-Micron Era. Interconnect Centric Design for Advanced SOC and NOC. Kluwer Academic Publishers, Netherlands, 2004.

[4] Benini, L.—De Micheli, G.: Networks on Chips: A New SoC Paradigm. Computer, Vol. 35, 2002, No. 1, pp. 70–78.

[5] Kahn, G.: The Semantics of a Simple Language for Parallel Programming. Proceedings of the IFIP Congress, Stockholm, Sweden, 1974, pp. 471–475.

[6] Lieverse, P.—van der Wolf, P.—Deprettere, E.—Vissers, K.: A Methodology for Architecture Exploration of Heterogeneous Signal Processing Systems. Proceedings of IEEE Workshop on Signal Processing Systems, Taiwan, 1999, pp. 181–190.

[7] Jalabert, A.—Murali, S.—Benini, L.—De Micheli, G.: XpipesCompiler: A Tool for Instantiating Application Specific Network on Chip. Proceedings of Design, Automation, and Test in Europe (DATE 2004), Paris, France, 2004, pp. 1530–1591.

[8] Ohring, S. R.—Ibel, M.—Das, S. K.—Kumar, M. J.: On Generalized Fat Trees. Proceedings of 9th International Parallel Processing Symposium. Santa Barbara, California, USA, 1995, pp. 37–44.

[9] Kariniemi, H.—Nurmi, J.: New Adaptive Routing Algorithm for Extended Generalized Fat Tree On-Chip. Proceedings of International Symposium on System-on-Chip, Tampere, Finland, 2003, pp. 113–118.

[10] Leiserson, C. E.: Fat Tree: Universal Networks for Hardware-Efficient Supercomputing. IEEE Transactions on Computers, Vol. C-34, 1985, No. 10, pp. 892–901.

[11] Ni, L. M.—Gui, Y.—Moore, S.: Performance Evaluation of Switch Based Wormhole Networks. Proceedings of the 9th International Parallel Processing Symposium, Santa Barbara, California, USA, 1995, pp. 37–44.

[12] Leiserson, C. E. et al.: The Network Architecture of the Connection Machine CM-5. ACM Symposium on Parallel Algorithms and Architecture, San Diego, USA, 1992, pp. 272–285.

[13] Chien, C.-K.—Scherson, I. D.: Self-Routing Least Common Ancestor Networks. Proceedings of the 4th Symposium on the Frontiers of Massively Parallel Computers, McLean, USA, 1992, pp. 513–514.

[14] Banikazemi, M.—Panda, D. K.—Stunkel, C. B.—Abali, B.: Adaptive Routing in RS/6000 SP-like Bidirectional Multistage Interconnection Networks. Proceedings of the 14th International Parallel and Distributed Processing Symposium, Cancun, Mexico, 2000, pp. 43–52.

[15] Kariniemi, H.—Nurmi, J.: New Routing Algorithm for Improving the Throughput of Fat Tree Interconnection Networks. Proceedings of International Conference on Computer Science and Technology, Cancun, Mexico, 2003, pp. 375–381.

[16] Ashenden, P. J.: Comparison of Recursive and Repetitive Models of Recursive Hardware Structure. Proceedings of VHDL International Users Forum, Oakland, California, USA, 1994, pp. 462–474.

[17] Dally, W. J.—Seitz, C. L.: Deadlock-Free Message Routing in Multiprocessor Interconnection Networks. IEEE Transactions on Computers, Vol. C-36, 1987, No. 5, pp. 547–553.

Heikki Kariniemi received the M.Sc. degree from Tampere University of Technology (TUT), Tampere, Finland, in 2000. Since then he has been working as a researcher at the Institute of Digital and Computer Systems at TUT. Before this he was working there as a research assistant. He has also been working for VLSI Solution Oy, Tampere, as an IC-designer before starting to work at TUT. He is currently working toward the Ph.D degree and his research topic is in the area of on-chip communication networks for future ULSI multiprocessor System-on-Chip circuits.

Jari Nurmi has been a professor at the Institute of Digital and Computer Systems at Tampere University of Technology, Tampere, Finland, since 1999. Before this he worked for VLSI Solution Oy, Tampere, as the company's Vice President responsible for development activities of DSP processors. His current research interests include system-on-chip integration, on-chip communication architectures, embedded and application-specific processor architectures, and circuit implementations for digital communication and DSP systems. He is also the author or co-author of over 80 international conference and journal papers, and co-editor of one book published by Kluwer.