Addressing Heterogeneous Bandwidth Requirements in Modified Fat-Tree Networks-on-Chips

A. Bouhraoua **M. E. Elrabaa**

Computer Engineering Department
King Fahd University of Petroleum and Minerals
P.O.Box 969, Dhahran, 31261
Saudi Arabia
Email: abouh,elrabaa@kfupm.edu.sa

***Abstract*— A novel approach for satisfying heterogeneous bandwidth requirements of clients connected using a modified Fat Tree network-on-chip is presented. The new approach allows the NoC designer to satisfy the bandwidth requirement of each client without the need to "overdesign" the NoC. Thus, the power can be reduced without impacting the performance of applications running on the NoC.**

***Index Terms*—Networks-On-Chip, Systems-on-Chip, ASICs, Interconnection Networks, Fat Tree, Routing**

I. Introduction

Technology scaling has enabled the integration of a higher number of processing elements, computation cores and memories. This resulted in an ever increasing computation and communication complexity of the design of *Systems-on-Chips* (SoCs). The very short time to market constraints has put more stress on the design methodologies. As a consequence, *Networks-on-Chips* (NoCs) paradigm has emerged as an alternative to ad-hoc wiring or bus-based global interconnect in SoCs [1-4]. The idea of making the NoC as just another particular, global, scalable IP block is very appealing to the time limitations imposed by the market requirements. Its main advantages are the fact that it provides a systematic solution for issues of compatibility, bandwidth requirements, and performance. Hence, the general consensus is that the communication requirements, as well as the design flow, of billion transistors SoC are best accommodated by shared, segmented interconnection networks [2-4].

There has been a significant amount of effort in the area of NoCs. The focus has mostly been on proposing topologies and routing strategies to implement NoCs. Recently, the trend has shifted towards engineering the solutions and providing design tools that are more adapted to the reality. For example, power analysis of NoC circuitry has intensively been studied[6][7]. More realistic traffic models have been proposed [9] and more adapted hardware synthesis methodologies have been developed.

This said, there is an aspect of NoCs that was quasi unanimously adopted and never discussed in papers. This aspect is the fact that all the router clients (or traffic generators) are considered to have the same bandwidth requirements. In other words, the client's link bandwidth is the same for all the clients of a given network. This is a serious limitation that has not been addressed adequately in the literature. Most papers discuss the issue from the traffic pattern perspective where the different traffic generators will have different bandwidth behaviors. Other contributions focus on reducing the hardware cost of the network using the static routing information[5]. The real issue is related to the network design itself. It is the answer to the question: why should we give a client more bandwidth than it will ever use just to meet the bandwidth requirements of another client.

This paper proposes a method that enables the generation of a class of NoCs, based on a modified Fat Tree topology that provides support for heterogeneous bandwidth requirements for clients. After this introduction, the Fat Tree topology, the network scaling method and the routing algorithm are all defined in section II. Section II provides a foundation for the understanding of the method presented in this work. The modified Fat Tree network is briefly presented in section III. Section IV presents the method for handling heterogeneous bandwidth requirements in the modified Fat Tree. Finally, conclusions are stated in section V.

II. Fat Tree Topology

The architecture considered is a new class of NoCs based on a sub-class of Multi-Stage Interconnection Networks topology (MIN). More particularly, a class of bidirectional folded MINs; chosen for its properties of enabling adaptive routing. This class is well known in the literature under the name of Fat Trees (FT) [7]. The FT has been enhanced by removing contention from it as detailed in [1].

A. Network Topology

The network is organized as a matrix of routers with n rows; labeled from 0 to $n-1$; and $2^{(n-1)}$ columns; labeled from 0 to $2^{(n-1)} - 1$. Each router of row 0 has 2 clients attached to it (bottom side). The total number of clients of a network of n

0-7695-3110-5/08 $25.00 © 2008 IEEE
DOI 10.1109/DELTA.2008.120

rows is 2^n clients. The routers of other rows are connected only to other routers. Any router is identified by its position (r, c); r denoting its row index and c denoting its column index.

In general, a router (r,c) is connected to two routers at row $r+1$:

- router $(r+1, c)$
- router $(r+1, c-2^r)$ or router $(r+1, c+2^r)$ depending whether the value $c/2^r$ is odd or even respectively

Figure 1 shows a regular FT of $3 \times (2^{3-1})$ routers and 2^3 clients. Router $(1, 2)$ is connected upwards to routers $(2, 2)$ and routers $(2, 0)$ because $2/2^1$ is odd. Router $(1, 2)$ is connected downwards to routers $(0, 2)$ and router $(0,3)$ because $2/2^0$ is even and $3/2^0$ is odd.

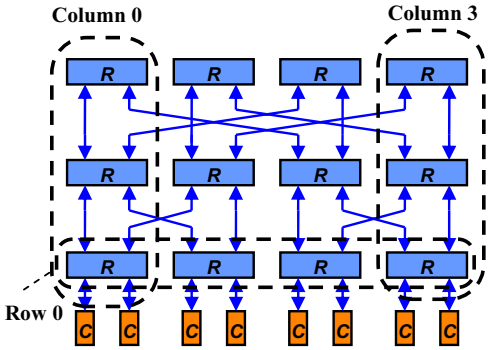


Figure 1: Regular Fat Tree Topology

B. Network Scaling

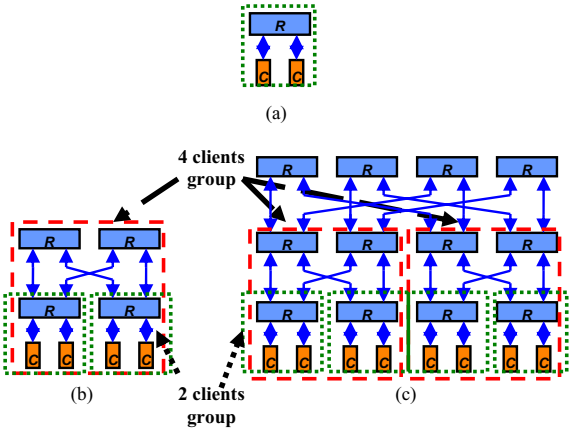
The FT network is built recursively starting from a single router with two clients attached o it. Figure 2 illustrates the recursive building of the network. It also shows that the routers and clients belong to groups. The notion of group will be useful later when describing the routing scheme.

A group of order r can be defined as follows (as illustrated in Figure 2):

- A structure of routers organized in r rows and 2^{r-1} columns.
- Routers at row 0 are always included in the group. Consequently, the clients, which are attached to routers at row 0, are also part of the group
- A group of order r is made of two adjacent groups of order $r-1$. Recursively, a group of order r contains 2^{r-1} routers columns and 2^r clients.

The number of groups of order r in the network is equal to 2^{n-r} since the total number of columns is 2^{n-1}.

Any router of coordinates (r, c) is connected to two adjacent groups of order r. The same router belongs to the group of order $r+1$ that contains the two groups of order r it is connected to. Any router at row r has its left link connected to the group of order r on the left and its right link connected to the group of order r on the right.



(a) 2 clients group (order 1);
(b) 4 clients group made of 2 two clients groups (order 2)
(c) 8 clients group made of 2 four clients groups (order 3)

Figure 2: Router Groups

C. Routing

Routing in FT simply follows the routing in binary trees. A packet is routed up until it reaches a router that has a path to its destination. This router is called routing *summit* for convenience. The FT structure, based on a superposition of binary trees, naturally provides packets with several upward paths. Any upward path will eventually lead to a summit where a downward path to the packet's destination is provided. The downward path to the packet's destination is unique as it is the case for any regular tree structure. Hence upward routing is adaptive while downward routing is deterministic.

Clients are labeled in an increasing order starting from the left to the right, i.e. all client addresses are within the interval $[0, 2^n - 1]$. The relation between the client address and the column coordinate of the router at row 0 attached to it is given by the following equation:

$$addr = 2*c + s \quad (1)$$

The constant s, $s = \{0, 1\}$, depends on the client's position. For the clients connected on the left of the routers at row 0, $s = 0$ and for those connected on the right, $s = 1$.

1) Reach Range

From the network scaling section, it has been established that any router (r, c) is connected to two groups of order r; a group G_L to its left with an associated address interval I_L and another to its right, G_R with a corresponding address interval I_R. These groups have the following properties:

- The size of I_L is equal to the size of I_R and is 2^r clients.
- G_L and G_R are adjacent because connected to router (r, c) thus I_L and I_R are contiguous intervals and $I_L < I_R$.

The two intervals I_L and I_R are hence computed as follows:

$$I_L = [P_L, P_L+2^r -1] \text{ and } I_R = [P_L+2^r, P_L + 2^{r+1} -1] \quad (2)$$

P_L corresponds to the address of leftmost client in G_L. Since P_L is an address, it can be written as (see equation 1):

$$P_L = 2*c_L + s_L$$

c_L, corresponds to the column coordinate of the leftmost router at row *0* of G_L. Since, P_L corresponds to the address of *leftmost* client in G_L, then $s_L = 0$ because client is on the left. Therefore:

$$P_L = 2*c_L,$$

Hence, equation (2) becomes:

$$I_L = [2c_L , 2c_L + 2^r -1] \quad \text{and} \quad I_R = [2c_L + 2^r, 2c_L + 2^{r+1} -1]$$

Any router or coordinates *(r, c)* connected to G_L and G_R has its column coordinate c vary within the interval $[c_L, c_L + 2^r - 1]$. This is because:

- The router *(r, c)* belongs to a group of order $r+1$ composed of the two groups G_L and G_R, which makes its lowest column coordinate the same as for group G_L.
- The number of router columns in this group is $2^{(r+1)-1} = 2^r$.

This actually means that c can be expressed as:

$$c = c_L + k;\ 0 \le k < 2^r.$$

The value k can be represented on r bits. Thus c_L is obtained by simply clearing the lowest r bits of the column coordinate c.

2) Finding the "Summit"

The summit is the first router reached by the packet that provides a path to the packet destination address *(daddr)*. This means that either one of the two intervals I_L and I_R; associated with the summit's left and right lower order groups will contain *daddr*.

Therefore, a router *(r, c)* is a summit if *daddr* $\in I_L$ or *daddr* $\in I_R$. The packet is routed to the left if *daddr* $\in I_L$, or to the right and if *daddr* $\in I_R$. Before reaching a summit, the packet is always routed up.

Once the summit is reached, the routing function will determine the path step by step at each stage when going downwards. The principle followed is to divide the interval that *daddr* belongs to (either I_L or I_R) to into two sub intervals and so on.

D. Modified FT

Within a single router, packet routing falls in one of two cases. The first case is the routing of packets coming from the bottom links. These packets are either routed up or routed down. The other case is for packets coming from the up links. These packets are always routed down. So, packets coming from the up links are never routed up. Only packets coming from the bottom links are routed up. Since the number of up links is equal to the number of bottom links, there cannot be any contention when routing up. Contention occurs when going down. Because the bottom links are split in right and left links, deterministic routing of packets will lead to contention. In other words, if several packets coming from the up links need to go right, there will be a contention. One of these packets will earn the right to use the link while the others will be waiting for it to complete as shown in Figure 3.

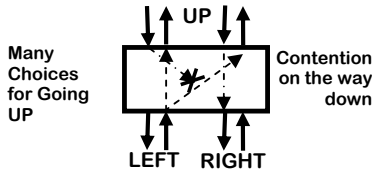


Figure 3 – Contention in FTs

Contention can be removed if there are enough output ports in a router so that they can accommodate any combination of incoming packets. Since Contention occurs only on the downward path, doubling the number of output ports in the downward direction will eliminate the contention. This is the adopted strategy for the proposed modified FT.

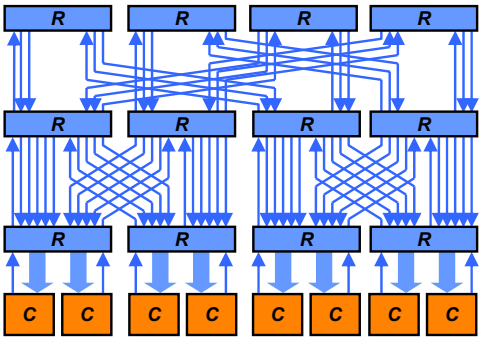


Figure 4 – Modified FT Topology

Figure 4 shows the modified FT where the down links are doubled. Doubling the output ports of a router also means doubling some of the input ports of the adjacent router, of lower stage, to which it is connected. This is needed to be able to connect all the output ports of the upper stage router. To avoid contention in the lower stage router, its output ports are twice as much as its input ports and four times the input ports of its upper stage adjacent router. The router architecture and performance evaluation have already been discussed in [1].

III. Addressing Heterogeneous Bandwidth Requirements

A. Problem Definition

To illustrate the issue of heterogeneous bandwidth requirements, we consider a network that has 5 clients that have the bandwidth requirements expressed in Table 1. The bandwidth requirement is defined as the maximum link speed required for transferring bursts and do not have a direct

relation with the traffic pattern of the client. The issue is for a regular NoC design, the maximum link speed of 800 Mbits/s will be chosen for all the clients. This constitutes a real waste of resource and power putting unnecessary stress on the already difficult timing closure task.

TABLE 1: EXAMPLE ON BANDWIDTH REQUIREMENTS

Client	Bandwidth Requirements (Mbits/s)	Unit Bandwidth
C_1	800	8
C_2	100	1
C_3	100	1
C_4	200	2
C_5	400	4

Another method would be to design a network with different speeds hence different clock domains. Because the crossing of many clock domains, the design requires cross boundary rate matching FIFOs which size depends on the traffic patterns of the different clients. Rate matching FIFOs will significantly increase the latency. The routers themselves need to be run at the highest frequency in order to avoid congestion due to rate difference. The clock domain crossings being only at the client-router interface annihilate any projected savings on power that can justify the existence of multiple clock domains. In summary, using multiple clock domains will create more problems than it will solve.

B. *Proposed Approach*

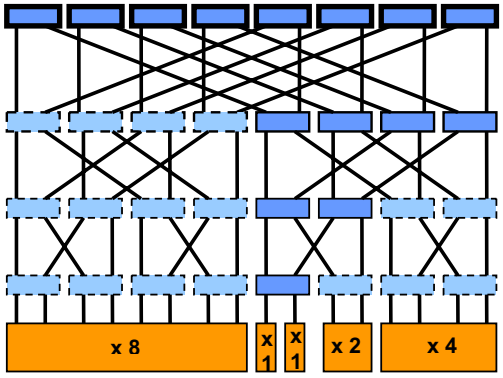


Figure 5 – Clients with Heterogeneous Bandwidth Requirements

The current proposal's corner stone rests on the permission for the clients to access the network through more than one port. To provide a client with a multiple of the unit link bandwidth, many links are reserved for the same client. The main advantage of this method is that the routers are clocked with the lowest frequency that corresponds to the unit link bandwidth. Figure 5 shows how the clients given in the example of Table 1 are connected to the network.

The approach is stated in what follows:

- Determine the bandwidth requirements for each client
- Express the bandwidth requirements as a multiple of a chosen unit bandwidth approximating the numbers.
- Add the units of bandwidth requirement
- Choose the closest higher power of *2* as number of clients to generate the modified Fat Tree.

C. *Routing Issues*

This method, as described so far, poses an issue with routing. Actually, all received packets by client C_1 will be only directed to the leftmost link, assuming that C_1 is labeled with address *0*. This means that the other links are totally idle. Ideally, packets from other clients should be delivered on any link assigned to client C_1. To enforce this behavior, no routing should be performed after packets destined to C_1 reach one of the routers on row *3* of the same group of order 4 C_1 belongs to (bold line routers on Figure 5). If no routing should occur after these locations, then there is no reason for keeping the routers. These ones are consistently removed from the network, as shown in Figure 6.

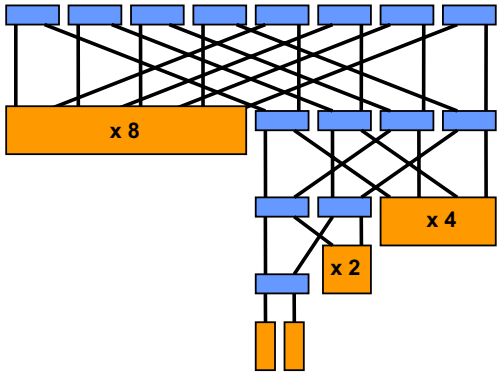


Figure 6 – Trimmed FT Network with Heterogeneous Bandwidth Requirements

All of the removed routers form in fact entire groups (dashed line boxes in Figure 5). Routers removed because of clients C_1, C_4 and C_5 form groups respectively of order *3, 1 and 2*. As the number of clients (ports) attached to a group of any order, as mentioned in section II, is always a power of 2, the number of ports that can be assigned to a single client is therefore a power of 2. A simple rule is to round bandwidth requirement factors to the nearest power of 2. Reducing the network components based on the clients communication static parameters is already becoming a trend in the research community [5].

```
client_address = 0;
for(i=0; i < clients_number; i++) {
  client[i] = client_address;
  client_address = client_address +
            client_bandwidth_req[i];
}
```

Figure 7 – Client Labeling Algorithm

D. Client Labeling

Labeling the clients becomes a little more complex as addresses should be carefully assigned to clients to avoid routing packets to the wrong destination. A client should be associated to a single address through which it can be reached and by which it is identified. However, all the addresses that fall within its routing space should not be used by any other client and should be reserved. For example, Client C_1 is labeled with address *0* and reserves all the addresses that fall in the interval *[1, 7]*. It does so because any packet which destination address is within this interval will be routed to one of the ports of C_1. This way, client C_2 will be labeled with address *8*, C_3 with address *9*, C_4 with address *10* and C_5 with address *12*. Addresses *[1,7]U[11]U[13,15]* are all reserved and cannot be used for packet transfer. The labeling process is trivial and follows a simple algorithm described in Figure 7 (written in pseudo-code).

It is important to stress the importance of the single address per client policy. Abandoning this policy will create problems of client identification and destination address dilemma for generating packets destined to a client that has more than one address.

E. Packet Injection/Reception Scheme

Packet injection/reception scheme is to be defined for the clients that have a bandwidth requirement greater than 1. Two solutions are possible for the packet scheme:

- The client considers all its input/output ports as independent ports. Packets are sent and received independently on any port. a packet is transmitted and received over a single port.
- The client considers all the ports as a single aggregated resource. Packets are divided into segments. Segments are sent and received in parallel over all the ports.

The first solution does not imply any modification of the existing architecture because packets continue to be as a single entity inside and outside the network. A major drawback, for this option, is that the use of a single port for sending an entire packet may force the client to reduce its packet generation rate loosing the justification for requiring more bandwidth. It is possible to overcome this drawback by designing an adequate client/network interface that has the capability of generating packets in parallel.

The second option is the segmentation of a single packet into small size segments. Each segment is considered as a packet by the network which routes it individually. A reassembly unit is in charge of putting the pieces back together on the receiver side. This method guarantees an optimal utilization of the available bandwidth as all the ports are involved in the packet transmission. The major drawback for this technique is the overhead generated by the implementation considerations. The packet header must include sequence number and packet ID fields to help reassemble the different segments. Because packet segments are received out of order, the delivery of the packet does not start until all the segments are received and accounted for. This requirement puts more stress on the space required by the receive FIFO memories on the receiver side of all the clients.

Both solutions have advantages and disadvantages. The Adoption of either one of these two solutions should be left to the application. For example, applications where clients have the capability of generating multiple packets simultaneously will favor the first solution whereas applications that have strict bandwidth requirements will opt for the second one.

IV. Conclusions

In this work, the possibility of supporting different bandwidth requirements for clients in NoCs was explored. A solution adapted to the modified Fat Tree topology, already presented in other communications, was presented. The solution provides a substantial benefit over the regular assumption of uniform bandwidth requirements. This advantage is the reduction the network operating clock frequency to match the client with minimum bandwidth requirement needs. Another fortunate side effect of this approach is the elimination of some of the network routing elements. The hardware cost is therefore reduced. However, some extra overhead is introduced because of the way packets are injected/received in the network. The quantitative evaluation of the overhead is going be addressed in the future.

Acknowledgment

This research project has been supported by King Fahd University of Petroleum and Minerals under project No FT-2006/25. This support is highly appreciated by the authors.

References

[1] A. Bouhraoua and Mohammed E.S. El-Rabaa, “A High-Throughput Network-on-Chip Architecture for Systems-on-Chip Interconnect,” *Proceedings of the International Symposium on System-on-Chip* (SOC06), 14-16 November 2006, Tampere, Finland.

[2] L. Benini and G. D. Micheli, “Networks on chips: A new SoC paradigm”, *IEEE Computer*, 35(1):70 – 78, January 2002.

[3] W. J. Dally and B. Towles. Route packets, not wires: On chip interconnection networks. In *Proceedings of the 38thDesign Automation Conference*, pages 684–689, June 2001.

[4] J. Henkel, W. Wolf, and S. Chakradhar, “On-chip networks: a scalable, communication-centric embedded system design paradigm,” in *Proc. 17th Int’l Conf. VLSI Design*, 2004, pp. 845-851.

[5] P. Meloni, S. Murali, S. Carta, M. Camplani, L. Raffo, G. De Micheli, “Routing aware switch hardware customization for Networks on Chips,” in *Proc. 1st Int’l Conf. NanoNets*, Sept. 2006, pp. 1-5.

[6] E. Nilsson and J. Öberg, “Reducing power and latency in 2-D mesh NoCs using globally pseudochronous locally synchronous clocking”, CODES+ISSS 2004.

[7] E. Nilsson and J. Öberg, “Trading off power versus latency using GPLS clocking in 2D-mesh NoCs”, ISSCS 2005

[8] C. Leiserson, "*Fat-Trees: Universal Networks forHardware-Efficient Supercomputing*", IEEE Transactions on Computers, vol. C-34, no. 10, pp. 892-901, October 1985.

[9] V. Soteriou, H. Wang and L. Peh, “A Statistical Traffic Model for On-Chip Interconnection Networks”, in *Proceedings of the 14th IEEE Intl. Symp. on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS '06)*, Sept. 2006, pp 104-116