Performance-Aware Design Of Dynamically Reconfigurable System-On-Chips With Quality Of Service Guarantees

[1]**Manjula M**, PG Scholar,
[2]**Mrs. J. Nalini**, Asst..Professor,
Department of Electronics and Communication Engineering,
PSN College of Engineering and Technology,
Tirunelveli, Tamil Nadu, India,

Abstract-Network-on-Chip (NoC) architecture has evolved as a promising solution and also provides an alternative to the bus-based communication mechanism that can meet the challenging requirements of performance, scalability and flexibility in a System-on-chip (SOC) based embedded design. The three major phases in a NOC design flow are scheduling, mapping and routing. This paper proposes a unified approach for solving the first two sub problems of NOC design rather solving them sequentially. Also the solution has been made scalable and generic to suit any random bench marks. In addition, a performance aware routing algorithm has been proposed to ensure quality of service guarantees with reduced time complexity. Thus the complete NOC design is addressed and the results show that time, delay and energy are improved compared to the previous approaches.

Index terms: Mapping, Network-on-chip, routing,Scheduling.

I INTRODUCTION:

Multicore architectures have become an emerging trend in the embedded systems technology and too prominent component to meet the functional and performance requirements of complex applications. These multi-core designs or system-on-chip besides offering high performance and flexibility ensures low cost and power efficient implementations. An important issue to be considered in improving the performance of these systems is the communication infrastructure available to interact between these many cores. Thus the design has evolved from a computation centric to a communication centric design. On Chip Networking proves to be a promising solution for such kind of design needs in the recent years. This Network-on-Chip (NOC) technology applies networking theory and methods to on-chip communication thus achieving notable improvements over the conventional bus and crossbar interconnects.

The conventional steps in the design flow of a NOC are task scheduling, core mapping and network routing. Task scheduling is the mapping of the application tasks to the processing elements of the system-on-chip. Core mapping deals with the placement of the processing elements on to the network tiles such that the communication latency and energy is utilized at its lowest possible level. Routing of data paths is the movement of network traffic on NOC architecture. This paper deals with the design of a NOC. The efforts have been made to make the proposed approach scalable to larger benchmarks and generic to all the architectures irrespective of its complexity. The main aim is to unify the steps involved in the design flow so as to obtain a more accurate result in the final design. The reason behind this unified approach is that the sequential approach proves to be faulty at times since the decision taken at an earlier phase may prove to have several disadvantages at a later phase. Thus to obtain better performance and lower the energy requirements, it is necessary to consider the scheduling and mapping as a single process.

II RELATED WORKS:

Many previous works addressed task scheduling and core mapping separately. The algorithms used were Genetic algorithm [1], Ant colony algorithm [2] and Integer Linear Programming [3]. Varatkar et al. [4] proposed a communication aware algorithm for task scheduling based on the inter-processor communication volume. Some works available on the unification process [5], [6] considered only regular mesh architectures. Ghosh et al. [7] used MINCOL algorithm for the unification problem but still that applies to regular mesh only.

III ORGANIZATION OF THE WORK:

This work proposes an approach to unify the processes of the design phase even with highly irregular architectures and also proposes a method to speed-up the entire process. Though the available methods[5] took into account only the first two phases of the process ,this paper guarantees

performance oriented design at every stage of the design by including a fuzzy logic and deadlock-free design in the routing phase too.

The methodology is organized as follows:

(i) Section-A explains a novel graph model which is used to calculate the energy spent on the communication of cores irrespective of the high irregularity of the network.

(ii) Section-B deals with a special heuristic employed to accelerate the entire process.

(iii) Section-C explains the MILP formulation and unification of the processes.

(iv) Section-D explains a performance aware routing algorithm for both congestion avoidance and deadlock free routing.

A. Latency graph:

Highly irregular NOC architectures are much common in the complex chips with highly complex functionalities. To cope up with the high network irregularity while calculating the energy and communication latency during core mapping, a brand new communication graph model called latency graph is proposed. This approach proves to have lesser time complexity than the previously available methods[8].

Steps to build a latency graph:

Problem formulation:

Let us consider a network topology represented by a graph G= {V, E} such that

V-> denotes the network tiles

E->denotes the energy consumption on each link

Procedure to build latency graph:

1. For any two nodes or network tiles, the minimal energy between V_i and Vj is calculated using Floyd-warshall algorithm.
2. Choose a root node and label it as '0'
3. Using Dijkstra algorithm, a new sub graph is formulated which defines the energy required to communicate from the tile chosen as the root node to all other tiles in the network.
4. If there is no node V_i € V such that it aids in finding the communication energy $Ecjk=(L_k^{(i)} - L_k^{(j)})$,then node V_i is chosen as the root node and the procedure is repeated from the previous step. This is continued until unless all the negative loops are eliminated and a finite solution is obtained.

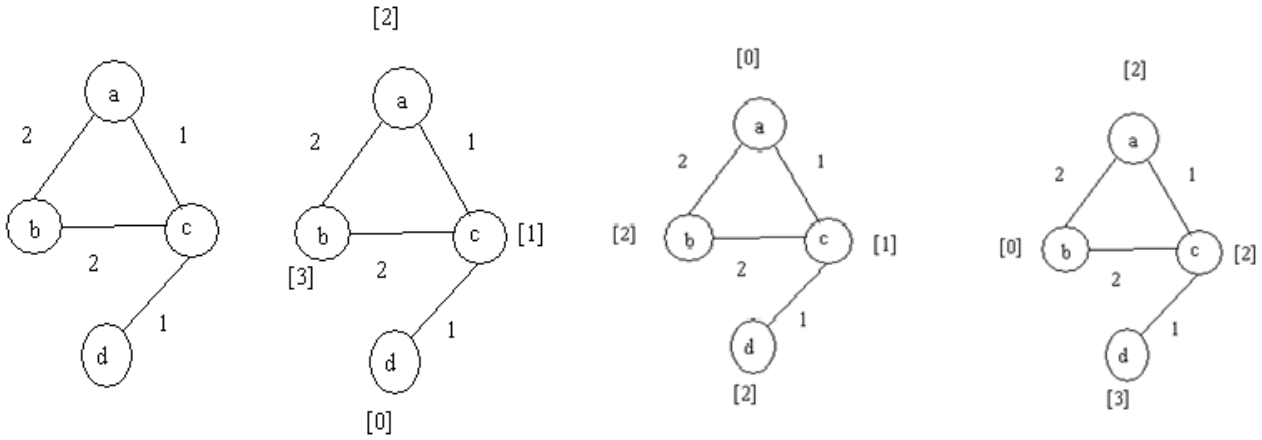
The example shown in fig 1 can be illustrated as follows:

Initially tile 'd' is picked up as the root node and the shortest path from node 'd' to all the other network tiles is calculated(fig 2).Then by making use of the latency cost available from this calculation, the energy required to communicate between any two cores is calculated by subtracting the labels available on those cores in the previously obtained sub graph. Since we find certain values in the third column (fig 2) are negative, it indicates that the energy is not available still in the calculation. Thus we start off with the tile 'a' and again all the values are calculated. This procedure will be repeated until no negative values are found. Finally, the maximum value among the available values is chosen as the communication latency. This is given by the theorem which states that among a set of available sub graphs of latency graph the communication energy is given by

E_{Cij} = max {available energies calculated through latency graph}

In the example, it is given by,

$E_{Cij} = \max\{ lat_j^{(d)} - lat_i^{(d)}, lat_j^{(a)} - lat_i^{(a)}, lat_j^{(b)} - lat_i^{(b)}\}$



(a) (b) (c) (d)

FIG 1: a,b,c,d – Steps in building latency graph

i->j	E_{Cij}	$lat_j^{(d)}$-$lat_i^{(d)}$	$lat_j^{(a)}$-$lat_i^{(a)}$	$lat_j^{(b)}$-$lat_i^{(b)}$	Max
a->b	2	1	2	-2	**2**
a->c	1	-1	1	0	**1**
a->d	2	-2	2	1	**2**
b->c	2	-2	-1	2	**2**
b->d	3	-3	0	3	**3**
c->d	1	-1	1	1	**1**
b->a	2	-1	-2	2	**2**
c->a	1	1	-1	0	**1**
d->a	2	2	-2	-1	**2**
c->b	2	2	1	-2	**2**
d->b	3	3	0	-3	**3**
d->c	1	1	-1	-1	**1**

Fig 2 : Table illustrating the values found through latency graph

the total number of latency sub graphs is constrained by the inequality[8]

$$2 \leq k \leq M$$

Where K->total number of latency sub graphs

B. Acceleration technique:

A task graph is the representation of the applications that are to be executed on a multi core system which consists of independent or dependent tasks of the applications. If the task graph is much larger to be executed on a SOC, then it will not be efficient to perform a search for a promising solution in the exhaustive space in terms of runtime complexity. Thus a partitioning approach which divides the larger task graphs into smaller sub graphs is utilized. The min-cut partitioning approach is explained as follows:

A threshold value is set for either partitioning the larger task graphs into smaller sub graphs or grouping the smaller graphs to a larger one. Then if the number of applications is greater than a threshold, they will be clustered to this threshold. On the other hand, if the number of the applications is less than the threshold, they will be partitioned to this threshold. The threshold value is set to

max (3,[M/4]) in this paper

Where M -> number of available cores

Then a set of cores and network tiles are related to a sub graph group depending on the nature of the application and the cores that are capable of executing it. Each task has a runtime denoted as r_{ij} associated with it when executed on a core. After partitioning the graph based on the mincut approach, the partitioned applications will be run on the core which consumes least run time compared to the other.

Steps in grouping cores and network tiles:

Fig 3 illustrates this min cut approach.

(i) Sort the task groups in ascending order by the number of sub-tasks in the application.

(ii) Find the cores that are capable of executing the tasks.

(iii) Construct a bipartite graph with communication volume within each group.

(iv) Find a weighted bipartite matching with least communication latency including network tiles for mapping.

C. Mixed integer programming formulation:

After partitioning the task graphs into smaller ones and calculating the communication energy using latency graph, these sub graphs along with cores capable of executing it and the energy spent on the computation and communication will be set as input to the MILP solver where we find a appropriate result for scheduling and mapping [3].Mathematically, a mixed-integer program is the minimization or maximization of a linear function subject to
linear constraints. Here, the variables involved in the function are

(i) Execution time of the task, t.

(ii) Total energy consumption which incorporates mathematically both inter core and intra core communication, E.

(iii) Total time margin before task deadline, d

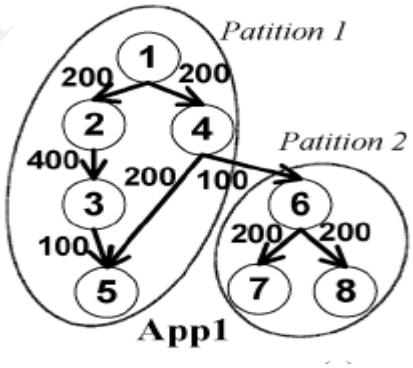


Fig 3 (a): A larger task graph

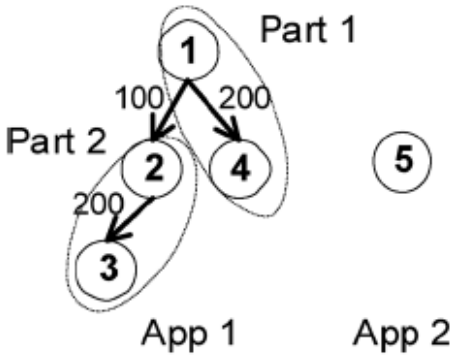


Fig 3 (b): Partitioned sub graphs of fig 3(a)

Thus the objective cost function to be minimized by the solver is

$$C=\alpha_0 T+\alpha_1 E+\alpha_2 d$$

$$=> \ C=\alpha_0 T+\alpha_1(E_{core}+E_{comm})+\alpha_2 t$$

Where C -> Cost function

α_0, α_1, α_2 -> weight factors assigned

Further the values of E_{core} and E_{comm} are

$$E_{core}=\sum_{0}^{M-1}\left(\sum_{0}^{N-1} X_{ij}W_{ij}P_j\right)$$

Where M -> total number of tasks
N -> total number of cores
P_j -> Power of the core j

$$E_{comm}=\sum_{(i,j)\in E} (C_{ij}e_{ij})$$

Where C_{ij} and e_{ij} ->communication volume and communication energy from task i to task j

D. Routing:

While modeling the communication latency ,it is necessary to incorporate the latency due to router's congestion. But it is hard to precisely estimate the congestion. Thus a heuristic based on fuzzy logic is employed to alleviate the congestion and assure the QOS guarantees thus making the approach performance aware. Complex applications such as Cloud computing, server consolidation, and real-time applications demand on-chip QOS support for security, performance isolation, and guarantees. Thus to avoid the problem of congestion and hotspots in Network on Chip a fuzzy logic control system is proposed. Fuzzy system normally substitutes or replace a skilled human operator with a rule-based system. The fuzzy logic uses linguistic descriptions to define the relationship between the input information and the output action with the help of an expert. In a network, the various metrics like collisions, traffic level and buffer occupancy are considered for congestion avoiding routing algorithm. First we have to fuzify the inputs or create membership values and put them into the fuzzy sets which are normalized in the range of (0, 1). The inference mechanism applies reasoning to compute fuzzy output. Fuzzification transforms the crisp value of the input variable into the fuzzy sets. It applies a predetermined set of linguistic rules shown in fig 5 and produces the fuzzy sets of the output linguistic variables. For the inference mechanism, the max-min method is used .Finally the defuzzification phase is done by using Center of gravity method to produce real numbers. The whole idea is shown in fig 4.

Deadlock occurs when two packets are waiting each other to be routed forward. Both of the packets reserve some resources and both are waiting each other to release the resources. Routers do not release the resources before they get the new resources and so the routing is locked. By finding neighbors and gathering available free slots buffer and considering this parameter as one of the input parameter for fuzzy controller, helps the proposed fuzzy routing algorithm to be deadlock free. It does not try to send a packet to a network path that is in the process of getting congested few hops after, but not yet propagated to the current switch[9]. By considering this parameter and crossbar demand as two input metrics and cost calculation by the fuzzy controller for each selective output channels, the output port with the minimum cost is selected.

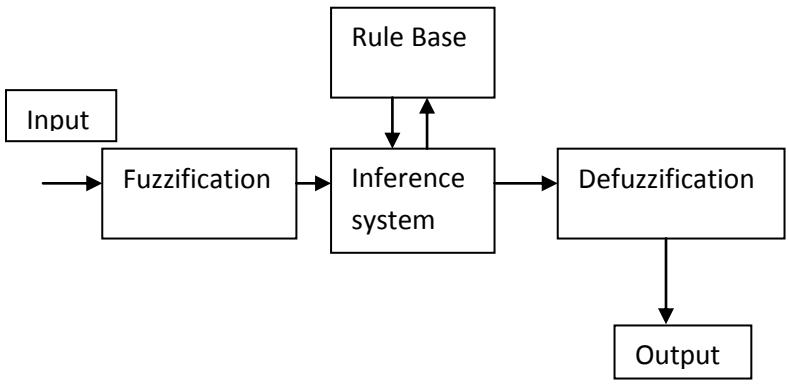


Fig 4: FUZZY Controller

Crossbar / Buffer	Low	Middle	High
Small	Medium	High	Very high
Medium	Low	Medium	High
Big	Very low	Low	Medium

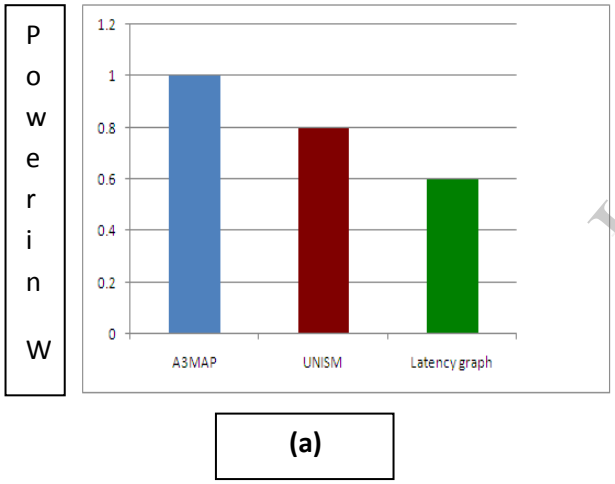
FIG 5: Fuzzy Rules

Results and Discussion:

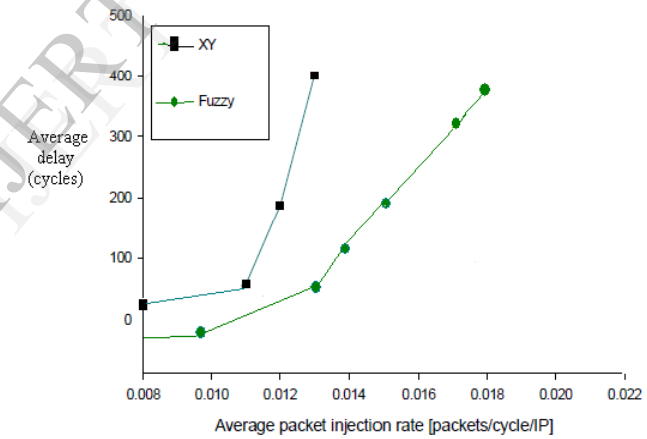
All the experiments were performed on an 3*3 irregular mesh. Mesh dimension 3*3 for MPEG4 and MWD applications and 4*4 mesh was used for heavy traffic scenarios. The parameters energy, time and delay are taken for test and the results were provided for different structures.

Conclusion:

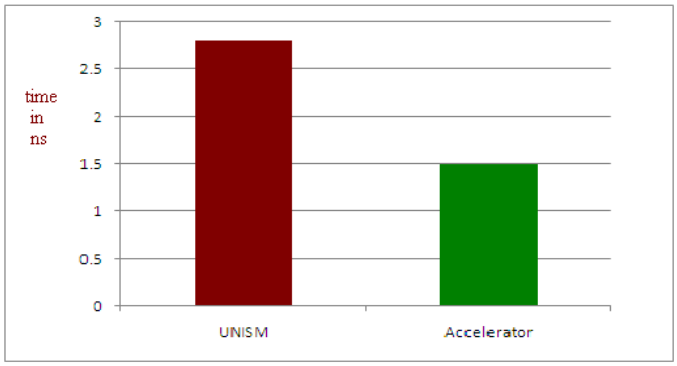
Thus an unified approach for scheduling and mapping is proposed. This utilizes a graph based approach which can cope with any degree of irregularity in the architecture. The QOS guarantees were provided with a performance aware design. The results show a 10% and 13.5% improvement in energy and QOS compare to the previous approaches.



(a)



(b)



(c)

Fig 6 a, b, c: Results showing power, delay reduction and increase in speed respectively

References:

[1] V. Kianzad, S. S. Bhattacharyya, and G. Qu, "CASPER: An integrated energy-driven approach for task graph scheduling on distributed embedded systems," in Proc. IEEE Int. Conf. Appl.-Specific Syst., Arch. Processors, 2005, pp. 191–197.

[2] P. C. Chang, I. W. Wu, J. J. Shann, and C. P. Chung, "ETAHM: An energy-aware task allocation algorithm for heterogeneous multiprocessor,"in Proc. Design Autom. Conf., 2008, pp. 776–779.

[3]C. Ostler and K. S. Chatha, "An ILP formulation for system-level application mapping on network processor architectures," in Proc. Conf.Design, Autom., Test Eur., 2007, pp. 99–104.

[4] G. Varatkar and R. Marculescu, "Communication-aware task scheduling and voltage selection for total systems energy minimization,"in Proc. IEEE/ACM Int. Conf. Comput.-Aided Design, 2003, pp.510–517.

[5]H. C. Chi, C. M. Wu, and J. H. Lee, "Integrated mapping and scheduling for circuit-switched network-on-chip architectures," in Proc. 4th IEEE Int. Symp. Electron. Design, Test, Appl. (DELTA), pp. 415–420.

[6] H. Yu, Y. Ha, and B. Veeravalli, Communication-aware application mapping and scheduling for NoC-based MPSoCs," in Proc. IEEE Int.Symp. Circuits Syst. (ISCAS), pp. 3232–3235.

[7]P. Ghosh, A. Sen, and A. Hall, "Energy efficient application mapping to NoC processing elements operating at multiple voltage levels," in Proc. ACM/IEEE Int. Symp. Netw. Chip, 2009, pp. 80–85.

[8] Ou He, Sheqin Dong, Wooyoung Jang, Jinian Bian and David Z. Pan ," UNISM: Unified Scheduling and mapping for General Networks on Chip"in Proc. IEEE Int conf. VLSI systems,2012.

[9] Ascia G, Catania V, Palesi M and Patti D (2008) Implementation and analysis of a new selection strategy for adaptive routing in network-on-chip. IEEE Transac. Comp. 57(6), 809-820.