Mapping and Configuration Methods for Multi-Use-Case Networks on Chips

Srinivasan Murali
CSL, Stanford University
Stanford, USA
smurali@stanford.edu

Martijn Coenen, Andrei Radulescu, Kees Goossens
Philips Research Laboratories
The Netherlands
{martijn.coenen,andrei.radulescu,kees.goossens}@philips.com

Giovanni De Micheli
LSI, EPFL
Switzerland
giovanni.demicheli@epfl.ch

Abstract

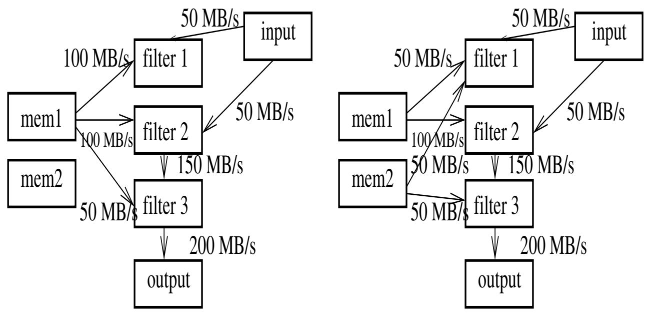
To provide a scalable communication infrastructure for *Systems on Chips* (SoCs), *Networks on Chips* (NoCs), a communication centric design paradigm is needed. To be cost effective, SoCs are often programmable and integrate several different applications or use-cases on to the same chip. For the SoC platform to support the different use-cases, the NoC architecture should satisfy the performance constraints of each individual use-case. In this work we motivate the need to consider multiple use-cases during the NoC design process. We present a method to efficiently map the applications on to the NoC architecture, satisfying the design constraints of each individual use-case. We also present novel ways to dynamically reconfigure the network across the different use-cases and explore the possibility of integrating *Dynamic Voltage and Frequency Scaling (DVS/DFS)* techniques with the use-case centric NoC design methodology. We validate the performance of the design methodology on several SoC applications. The dynamic reconfiguration of the NoC integrated with DVS/DFS schemes results in large power savings for the resulting NoC systems.

Keywords: Systems on Chips, Networks on chips, Use-Cases, Multiple application platforms, Dynamic, Reconfiguration, Voltage Scaling, Frequency Scaling, Guaranteed Throughput, Best Effort.

I. Introduction

As the number of transistors on a chip increases with every technological generation, the number of processor, memory and hardware cores available on the chip also increases. Thus, functionalities that were carried out by several different chips are being integrated on to a single chip, forming a *Systems on Chip (SoC)*. This, coupled together with the increase in the operating speed of the transistors has created the availability of large computational power for such systems. The challenges facing the SoC designer are to efficiently tap the available computational power under tight power budgets and meet the tight time-to-market constraints.

As the computational loads on the SoC increases, so does the load on the communication architecture. To support the high communication needs of multi-application SoC platforms, scalable on-chip interconnection networks are needed. A communication centric design paradigm, *Networks on Chips (NoCs)*, has been presented to address the interconnect issues of SoCs [2]-[6]. NoCs provide a scalable communication infrastructure with structured and modular wiring between the components. NoCs also help meet the tight time-to-market constraints, as the scalable architecture can be re-used across multiple platforms.



(a) Use-case 1 (b) Use-case 2

Fig. 1. A fragment of communication for two different use-cases of a set-top box SoC

To be cost effective, SoCs are often programmable and integrate several different applications or use-cases on to the same chip. As an example, a SoC for a set-top box has multiple resolution video processing capabilities (like high definition, standard definition), multiple picture modes (like split-screen, picture-in-picture), video recording features, high speed internet access and file transfer services, etc. [9]. Such convergence of multiple use-cases on to the same device is being observed in other electronic devices as well, such as the cell-phone or the personal digital assistant.

The different use-cases run on the SoC, although share many of the hardware components, could have very different performance requirements and design constraints for the communication architecture. As an example, we consider a simplified version of a SoC used in television set-top boxes [9], with support for four different use-cases. The communication bandwidth requirements for some of the connections between the components of the SoC for two of the use-cases are shown in Figure 1. Although we want the NoC to support all the use-cases, a NoC that is designed to run exactly one use-case does not necessarily meet the design constraints of the other user cases. In many of the existing NoC design methods, the NoC is designed and optimized for a single use-case or for a single application-trace of the design [10]-[15]. Such a trace based approach captures the characteristics and constraints of a single use-case very well, but fails to capture the multiple use-case scenario. Such a method averages out the communication effects across all the use-cases, which may result in a design that is unacceptable for many use-cases. As an example, when such a method is applied to perform NoC mapping for the set-top box SoC, the resulting NoC violates the design constraints of all the four use-cases.

Today's high-end SoCs support several hundred use-cases and manually checking whether the design constraints of the individual use-cases are satisfied by the NoC is a tedious process. Moreover, if the NoC design for the use-cases is carried out individually, it is difficult to converge to a single NoC design that satisfies the design constraints of all the use cases. In this work we motivate the need to consider the design constraints of the individual use-cases during the NoC design process. We present a design method for mapping of cores on to the NoC, considering the NoC configuration (i.e. path selection and resource reservation in the NoC) as sub-problems during the mapping phase, such that the resulting design satisfies the constraints of all the use-cases of the SoC. We then present methods to decrease the required network resources by dynamically reconfiguring the network across different use-cases. We also explore the effect of DVS/DFS techniques for reducing the power consumption of the network across the different use-cases. The methods are validated by performing experiments on several SoC designs.

II. Previous Work

Several researchers have proposed different architectures and design methodologies for the switches, links and Network Interfaces (NI), which are the major components of a NoC [18, 7, 20, 8]. Design flows that automate many of the steps of the design process have been presented in [17, 19]. In [8], the *Æthereal* architecture that supports Quality-of-Service (QoS) for applications by using Guaranteed Throughput (GT) connections for traffic streams that has bandwidth/latency constraints and by using Best Effort (BE) connections for the remaining traffic streams is presented.

The topology selection process and mapping of applications on to NoC architectures have been explored by many researchers. In [10, 11], branch-and-bound algorithms to map cores on to a mesh NoC topology for different routing functions are presented. In [12, 13], design methods and tools for mapping applications on to regular NoC topologies and automating the topology selection process has been presented. In [16], the methods are extended to consider the QoS constraints during the mapping phase. Building application specific buses and NoC topologies has been presented in [21, 14]. In [15], a tool that automates the combined mapping and NoC configuration steps for the *Æthereal* is presented. In all these NoC design works, the design methods assume a single set of communication constraints, which is obtained either for a single application or is obtained from a single trace for multiple applications.

In the RAW chip-multiprocessor, the interconnection network connectivity is reconfigured with the assistance of the compiler [23]. In the FLEXBUS architecture [22], the authors present methods to dynamically remove the overhead of bridges in multi-bus communication and provide methods where a core can be connected to different buses dynamically.

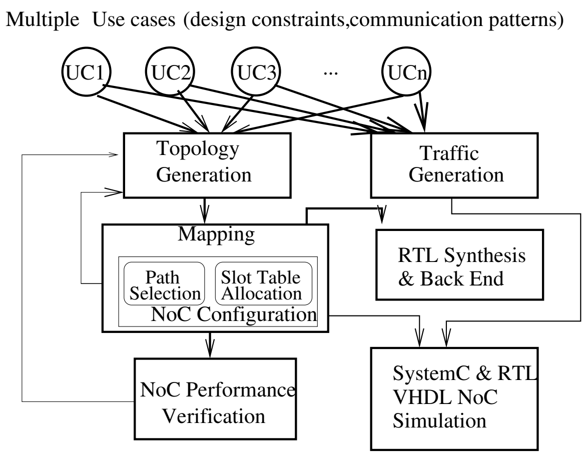
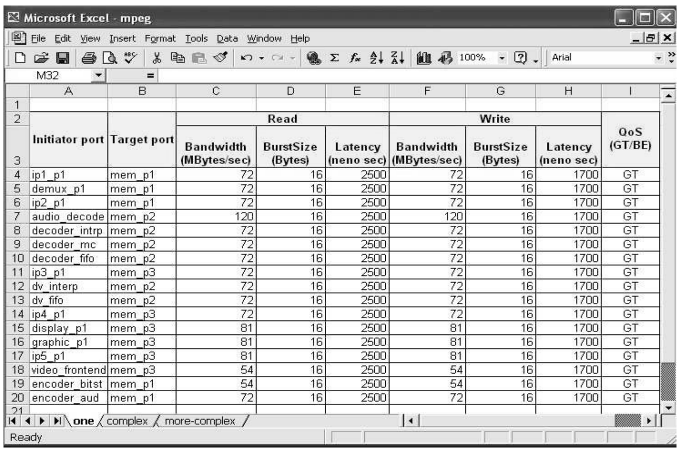


Fig. 2. Design Flow for NoCs



	A	B	C	D	E	F	G	H	I
1									
2	Initiator port	Target port	Read			Write			QoS (GT/BE)
3			Bandwidth (MBytes/sec)	BurstSize (Bytes)	Latency (neno sec)	Bandwidth (MBytes/sec)	BurstSize (Bytes)	Latency (neno sec)	
4	ip1_p1	mem_p1	72	16	2500	72	16	1700	GT
5	demux_p1	mem_p1	72	16	2500	72	16	1700	GT
6	ip2_p1	mem_p1	72	16	2500	72	16	1700	GT
7	audio_decode	mem_p2	120	16	2500	120	16	1700	GT
8	decoder_intrp	mem_p2	72	16	2500	72	16	1700	GT
9	decoder_mc	mem_p2	72	16	2500	72	16	1700	GT
10	decoder_fifo	mem_p2	72	16	2500	72	16	1700	GT
11	ip3_p1	mem_p3	72	16	2500	72	16	1700	GT
12	dv_interp	mem_p2	72	16	2500	72	16	1700	GT
13	dv_fifo	mem_p2	72	16	2500	72	16	1700	GT
14	ip4_p1	mem_p3	72	16	2500	72	16	1700	GT
15	display_p1	mem_p3	81	16	2500	81	16	1700	GT
16	graphic_p1	mem_p3	81	16	2500	81	16	1700	GT
17	ip5_p1	mem_p3	81	16	2500	81	16	1700	GT
18	video_frontend	mem_p3	54	16	2500	54	16	1700	GT
19	encoder_bitst	mem_p1	54	16	2500	54	16	1700	GT
20	encoder_aud	mem_p1	72	16	2500	72	16	1700	GT
21									

Fig. 3. Example input file with design constraints for an MPEG application

III. The Use-case Centric Design Flow

In this section we present the NoC design flow with the support for multiple use-cases integrated in to the flow (Figure 2). The NoC design flow and the mapping algorithms for the NoC for a single use-case were presented in [15, 17]. In this work, we extend the tool chain to support the multiple use-case scenario that is commonly encountered in SoCs. The communication design constraints for the different use-cases of the SoC are input to the design flow in the excel and xml file formats. The communication design constraints for each use-case includes the required bandwidth for various connections between the cores in the use-case, the maximum latency allowed for the connection, the QoS level required for the connection (like GT or BE), etc. An example fragment of the input file is presented in Figure 3.

With the different use-cases as input, in the first two phases of the design flow, the topology exploration and mapping of the use-cases on to the NoC are performed. The NoC configuration phase in which path selection and TDMA slot-table allocation (required for the GT traffic) are performed, is integrated with the mapping phase. The RTL level VHDL and SystemC models for the resulting NoC configuration are then automatically generated, which can then be simulated. The performance of the NoC can also be verified in parallel by the automatic performance verifier, which analytically checks whether the design constraints are met. The extension of the tool chain to support multiple use-cases is performed in a modular fashion without affecting most of the existing flow. As the multi-use-case NoC design methods are integrated with the tool chain, performance validations of the methods can be easily carried out to analyze the efficiency of the design methods.

IV. The Mapping Algorithm

In this section, we first present the mapping algorithm for a single use-case and then present methods to extend the algorithm for multiple use-cases.

A. Mapping Algorithm for single use-case

The mapping algorithm for a single use-case is presented in detail in [15]. In this sub-section, we only present a brief version of the algorithm highlighting the major phases. As in general, graph mapping is a NP-Hard problem [10, 13], a heuristic algorithm is used to perform the mapping. The selection of paths for the different traffic flows and the reservation of TDMA slot-table entries for the GT traffic flows are unified with the mapping process. The mapping algorithm is presented in Algorithm 1. At the outermost level of the algorithm, a NoC topology is generated. In the outer loop, the size of the topology is increased until a feasible mapping is obtained in the subsequent phases. Initially, all the cores of the SoC are unmapped. In the first step of the mapping algorithm, the traffic flows between the communicating cores are sorted in a non-increasing order. Then for each flow in the order, the source and destination cores of the flow, if they are not already mapped, are mapped on to the NoC. When performing the mapping of these cores, the path with the least cost that satisfies the bandwidth and latency constraints for the flow is chosen and the cores are mapped to the NIs in the path. A path is assumed to originate from a NI, traverse one or more switches and terminate in a NI. The cost of the path is a combined metric that considers an affine combination of the latency and bandwidth requirements for the flow. The slot-table reservation for the flow is also carried out in this step. The procedure is repeated for all the flows in the SoC. The approach also takes in to account the possibility of multiple cores sharing a single NI for communication. Note that once the initial mapping step is performed, the solution space can be explored by considering swapping of vertices using simulated annealing or tabu search, as performed in [16].

Algorithm 1 Mapping Algorithm for a single use-case

```
OUTERLOOP: Generate a NoC topology.
  1. Sort the traffic flows between the cores in a non-increasing order of the bandwidth requirements.
  2. For each flow in order:
    a. Choose a least-cost path for the flow that satisfies the bandwidth, latency constraints.
    b. If the source or destination cores of the flow are not yet mapped, map them on to the NIs in the path.
    c. Reserve the required bandwidth across the ports and reserve the slot-table entries for the flow.
If the resulting mapping violates design constraints, increase the size/resources of the topology and go to OUTERLOOP.
```

We refer the interested reader to [15] for the time complexity of the algorithm, details of path selection, other optimizations carried out and for the performance evaluation of the algorithm for several SoC designs.

B. Mapping Design Approach for Multiple Use-cases

When the SoC has multiple use-cases, we assume that all of the use-cases utilize the same mapping of cores on to the NoC components. This is because, if each individual use-case has a different mapping, then each core potentially needs to be connected to several different NIs, which may not be feasible because of physical layout restrictions and wiring complexity.

A direct extension of the single use-case mapping algorithm to support multiple use-cases would be to perform the mapping for the most communication intensive use-case and reuse the mapping for the other use-cases. However, as the design constraints of the use-cases can be very different, such a method may result in a mapping that does not satisfy the performance constraints of many of the use-cases. As an example, when such an approach is applied to the SoC considered in section I, the resulting NoC design satisfies only 2 of the 4 use-cases.

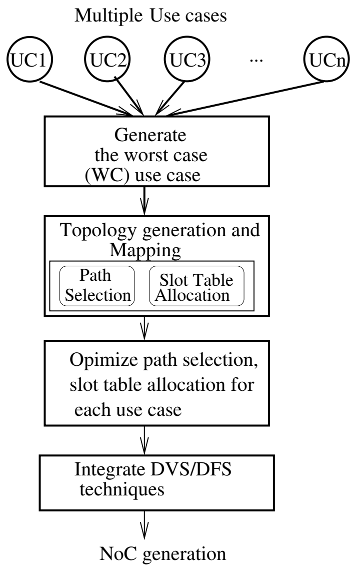
We use the following design method to extend the mapping design procedure for multiple use-cases (Figure 4(a)). In order to obtain a mapping that satisfies all the use-cases, we construct a synthetic Worst-Case (WC) use-case from the given set of input use-cases. For the communication flow between every pair of cores, the maximum required bandwidth values and the minimum required latency values for the flow across all the use-cases are selected and used in the WC use-case. A small example is presented in Figure 4(b). Thus the design constraints of all the individual use-cases are subsumed in the WC use-case and any NoC design that satisfies the constraints in the WC use-case will satisfy the constraints of each individual use-case. The WC use-case is then used for the mapping process. Due to the manner in which the WC use-case is constructed, the selected paths and slot-table allocations from the mapping process will satisfy the design constraints of each individual use-case.

Once the mapping is obtained from the WC use-case, we perform an optimization step, where we fix the mapping that is obtained from the WC use-case, but re-run the path selection and slot-table reservation phases individually for each use-case. We perform this for two reasons. First, the WC use-case had the worst case constraints from each use-case and by re-running the path selection and slot-table reservation steps and choosing the maximum values from the individual use-cases we can reduce the NoC resources, while still satisfying the constraints of all the use-cases (experimental evidence presented in Section VI A). Second, when the configuration time between the use-cases is large, the frequency and voltage of operation of the NoC can be scaled to match each individual use-case, which can result in significant power savings for the system.

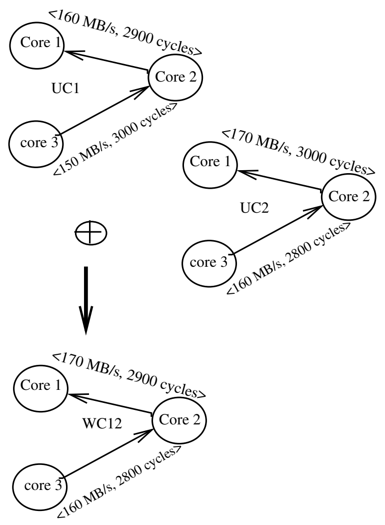
In general, when the paths and slot-tables used by the different use-cases are different, we need mechanisms to store them in memory and load them on to the network dynamically or compute them on the fly for the use-case. This is explored in the next section.

V. Dynamic Reconfiguration of the NoC

For most SoC designs, when the system switches between use-cases, some configuration time is needed for loading the


Multiple Use cases
UC1
UC2
UC3
...
UCn
Generate the worst case (WC) use case
Topology generation and Mapping
Path Selection
Slot Table Allocation
Opimize path selection, slot table allocation for each use case
Integrate DVS/DFS techniques
NoC generation

(a) Multi use-case mapping design flow



(b) Example of WC use-case constraints generation

Fig. 4. Multi use-case mapping flow and WC use-case generation

new use-case. This is mostly attributed for loading the use-case data and code, sending control signals to different parts of the design and for gracefully shutting down the already running use-case. In many designs, this use-case switching time is of the order of few milli-seconds. This configuration time can be utilized by the NoC for switching to a different path and slot-table allocation for the mapping. This time delay can also be utilized to vary the clock frequency/voltage of the NoC to match the use-case performance level.

In the *Æthereal* architecture [8], a static path routing scheme is used, where the paths are selected at the source NI of the traffic flow. Thus, the NIs maintain the path and slot-tables for the various connections. When the paths and slot-tables used by the NIs vary across different use-cases, the tables need to be stored in memory. As the on-chip memory available is mostly limited and as the use-case switching time is large, we use the off-chip memory to store the paths for the different use-cases.

We investigated the overhead for the reconfiguration mechanism for the set-top box SoC. The amount of data required to store the path and slot-table information for each use-case is around 560 Bytes. With 4 use-cases, the memory requirement for the reconfiguration mechanism is 2.24 KB. The time required to load the data from the memory and spread it around the NoC for an use-case is of the order of micro-seconds and the energy dissipation is of the order of micro-Joules. Using traditional mechanisms to scale the frequency and voltage of the system may require few milliseconds for configuration. Thus we can envision three different ways of NoC operation. First, when the use-cases that run on the SoC switch very frequently or when the initial configuration times are not acceptable (as in real-time use-cases), the different use-cases can use the WC use-case configuration. In this configuration, all the use-cases will use the same set of paths and slot-table allocations, thereby not requiring the NoC to be re-configured when the use-cases switch, resulting in seamless switching between the use-cases. However, this leads to an over-design of the network when compared to the scenario where the NoC is re-configured to suit the individual use-cases. Second, when the use-case switching is not that frequent, the NoC configuration (path and slot-tables) can be changed dynamically across use-cases, leading to a smaller NoC design (in terms of network components or frequency of operation). Third, when the use-cases are expected to run for a long time, the voltage or frequency of operation of the NoC can be varied to match the use-cases, resulting in large power savings for the system. The simulation results for these cases are presented in the next section.

VI. Simulation Results

We present simulation results on applying the multi-use-case design procedure on to 4 different SoC designs: P1 (with 2 use-cases), P2 (2 use-cases), P3 (4 use-cases) and P4 (8 use-cases). The designs P1-P3 are simplified versions of set-top box SoCs [9] and the design P4 is a video processing SoC used in TVs. Each use-case has a large number of (50 to 150) communicating pairs of components. A fragment of two of the use-cases used in the P3 design was presented earlier in Figure 1. The set-top box SoCs and the TV processor have different functionalities and communication patterns. The designs P1-P3 use an external memory for storing and retrieving data and the amount of data communicated to the memory is very large when compared to the rest of the design. The P4 design uses a streaming architecture with local memories on the chip, there by distributing the communication load across several components. We apply our design method to these SoCs with different architectures to validate the generality of the methods.

A. Effect of Mapping on the NoC Frequency

To evaluate the mapped designs, we fixed the topology and the maximum slot-table size for each design and we found the

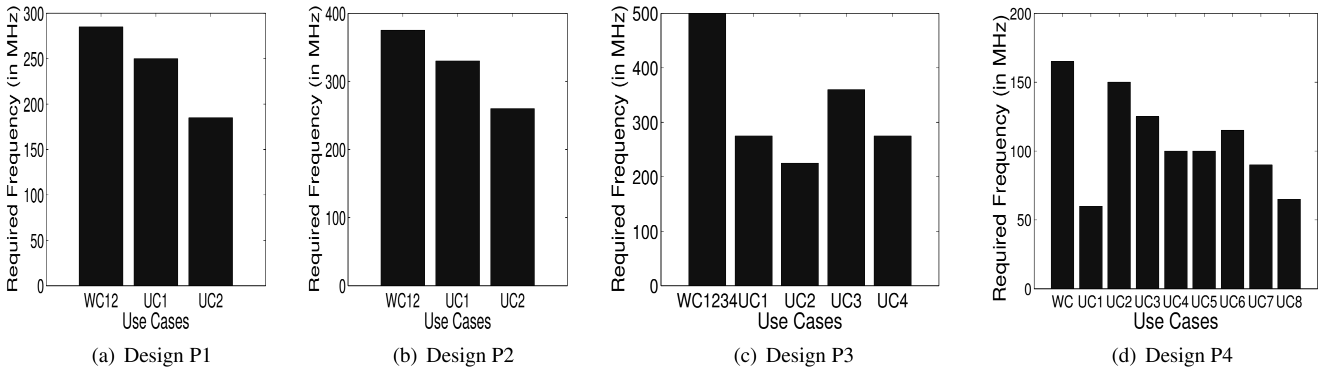


Fig. 5. The NoC operating frequencies required to support the different use-cases for the various designs. The WC use-case values are obtained when the path selection and the slot-table reservation are based on the WC use-case. The other values show the effect of re-applying the path selection and slot-table reservation for each use-case with the mapping obtained from the WC use-case.

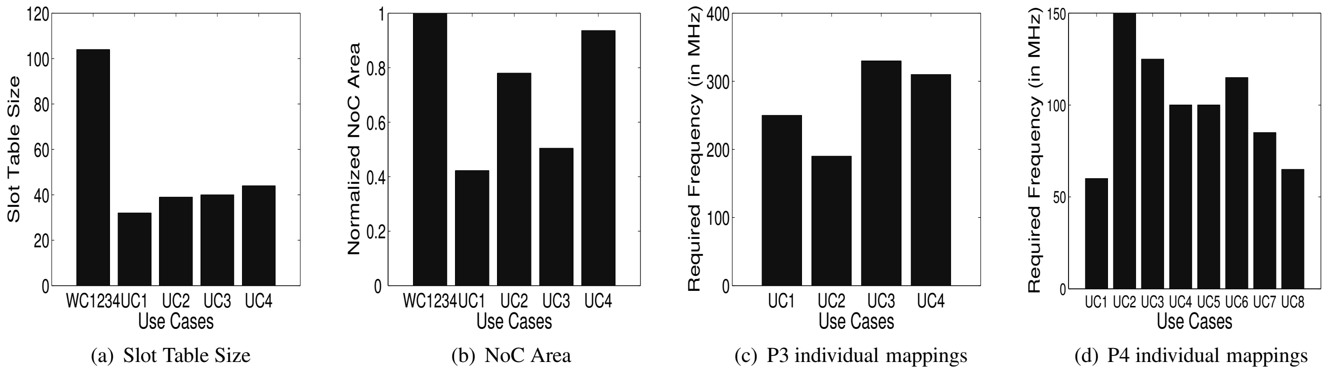


Fig. 6. (a)-(b) The effect of the mapping design procedure on the slot-table size and NoC area, (c)-(d) NoC frequency requirements for individual mapping of use-cases in designs P3 and P4.

minimum frequency of operation required by the NoC to support the different use-cases. The results of the mapping procedure for the 4 SoC designs are presented in Figure 5.

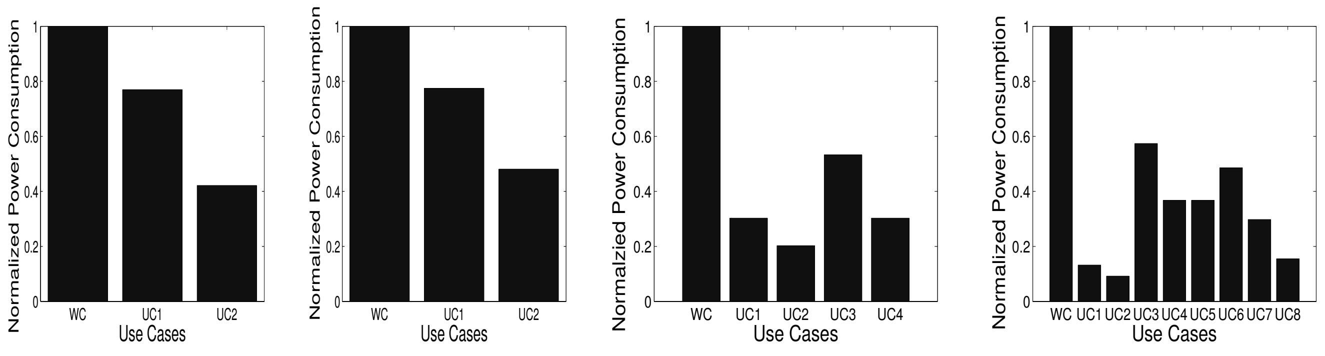
The frequency of operation required for the WC use-case is obtained from applying the NoC configuration (i.e. the path selection and the slot-table reservation) procedure on the WC use-case. The frequency of operation of the NoC after re-applying the configuration procedure for each of the use-cases, fixing the mapping from the WC use-case is also presented in the figures. When DVS/DFS techniques are not used and a single frequency of operation is used for all the use-cases, we need to take the maximum of the frequencies of each of the individual use-cases as the operating frequency of each design. In this case, re-applying the NoC configuration step results in 9% to 38% reduction in the required NoC operating frequency across the different designs. A lower operating frequency implies lower power consumption and smaller impact of noise sources.

In the above analysis, we assumed that the slot-table size is fixed and the NoC frequency is varied to support the use-cases. We also explored the effect of fixing the NoC frequency (at 500 MHz) and varying the slot-table size. As similar results were observed for all the designs, we only present the results for the P3 design (Figure 6(a)). We obtain 58% reduction in the slot-table size for the design by re-applying the NoC configuration step. A smaller slot-table size usually corresponds to a lower area for the NoC and lower packet latencies (as the traffic streams wait lesser to get the slots). The NoC area reduction due to the slot-table reduction (Figure 6(b)) is 10% for this design (the NoC area includes the area of the switches and the network interfaces). Trade-offs involving the frequency savings and area savings can also be explored.

B. Comparisons with the individual use-case mappings

To evaluate the optimality of the NoC design produced by the above method, we performed individual mappings for each of the use-cases in the P3 and P4 designs. The required NoC frequencies for the use-cases in the resulting designs are presented in Figures 6(c) and 6(d). These frequency values are the minimum possible values for the use-cases, as we have done the mappings individually and they provide a lower bound on the quality of solutions that can be obtained when all the use-cases share the same mapping[1]. When the results from Figures 6(c) and 6(d) are compared with the results from Figures 5(c)

[1]Note that the heuristic nature of the mapping algorithm can sometimes invalidate this general statement.



(a) Design P1 (b) Design P2 (c) Design P3 (d) Design P4

Fig. 7. Effects of DVS/DFS.

and 5(d), we find that the multi-use-case mapping design procedure results in mappings that require operating frequencies that are with in 10% of the minimum possible operating frequencies. We also performed experiments fixing the frequency of operation for the multi-use-case mappings to be the same as the individually mapped designs and varied the network resources needed to support all the use-cases in the designs. The multi-use-case mappings required slightly more resources (1%-10% increase in the NoC area) to support the same frequency of operation as the individual mappings.

C. Effect of DVS/DFS

When the frequency of the NoC is scaled to match the frequencies required for the individual use-cases, large power savings can be achieved. As the frequency of the network is scaled, the supply voltage required for operation can also be scaled to match the frequency. We use a conservative model for voltage scaling, where we assume that the square of the voltage scales linearly with the frequency [24]. The power savings achieved by the DVS/DFS techniques for the entire SoC platform depends on the amount of time each use-case is expected to run. Thus, in this experiment, we present the power savings achieved for each use-case of the platform separately. The power consumption of each of the use-cases, normalized with respect to the power consumption of the WC use-case is presented in Figure 7. On average, we obtain 59.21% power savings by using the DVS/DFS techniques across the different use-cases for the designs.

VII. Conclusions

As the number of applications or use-cases integrated on to a single SoC increases, the designer is faced with the challenge of building an interconnect structure that supports the design constraints of all the use-cases. In this paper we motivated the importance of the problem and presented use-case centric design methods to map applications on to NoC architectures. We also presented a way to dynamically configure the interconnect to support multiple use-cases and integrated Dynamic Voltage and Frequency (DVS/DFS) techniques with the reconfiguration mechanism. In future, we plan to extend the algorithms for supporting concurrent operation of use-cases and apply the use-case models for addressing other NoC design issues such as the application specific topology design.

References

[1] R. Ho et al., "The Future of Wires", Proceedings of the IEEE, April 2001, pages 490-504.

[2] L. Benini and G.De Micheli, "Networks on Chips: A New SoC Paradigm", IEEE Computers, pp. 70-78, Jan. 2002.

[3] W. J. Dally, B. Towles, "Route packets, not wires: on-chip interconnection networks", Proc. DAC 2001.

[4] D.Wingard,"MicroNetwork-Based Integration for SoCs", Design Automation Conference DAC 2001, pp. 673-677, Jun 2001.

[5] F.Karim et al., "On-chip communication architecture for OC-768 network processors", Design Automation Conference, pp. 678-678, June 2001.

[6] M. Sgroi et al. , "Addressing the System-on-a-Chip Interconnect Woes Through Communication-Based Design", Proc. DAC 2001.

[7] P.Guerrier, A.Greiner,"A generic architecture for on-chip packet switched interconnections", DATE 2000, pp. 250-256, March 2000.

[8] E. Rijpkema et al., "Trade offs in the design of a router with both guaranteed and best-effort services for networks on chip", DATE 2003.

[9] S. Dutta et al., "Viper: A multiprocessor SOC for advanced set-top box and digital TV systems", IEEE Design and Test of Computers, pages 21-31, Sept-Oct 2001.

[10] J. Hu, R. Marculescu, "Energy-Aware Mapping for Tile-based NOC Architectures Under Performance Constraints", Proc. ASP-DAC 2003.

[11] J. Hu, R. Marculescu, "Exploiting the Routing Flexibility for Energy/Performance Aware Mapping of Regular NoC Architectures", Proc. DATE 2003.

[12] S. Murali, G. De Micheli, "Bandwidth-Constrained Mapping of Cores onto NoC Architectures", Proc. DATE 2004.

[13] S. Murali, G. De Micheli, "SUNMAP: a tool for automatic topology selection and generation for NoCs", Proc. DAC 2004.

[14] A. Pinto et al., "Efficient Synthesis of Networks On-Chip", Proc. ICCD, 2003.

[15] A. Hansson et al., "A unified approach to constrained mapping and routing on network-on-chip architectures", pp. 75-80, Proc. ISSS 2005.

[16] S. Murali et al., "Mapping and Physical Planning of Networks-on-Chip with Quality-of-Service Guarantees", Proc. ASPDAC 2005.

[17] K. Goossens et al., "A Design Flow for Application-Specific Networks on Chip with Guaranteed Performance to Accelerate SOC Design and Verification", pp. 1182-1187, DATE 2005.

[18] S.Kumar et al., "A network on chip architecture and design methodology", ISVLSI 2002, pp.105-112, Apr 2002.

[19] D. Bertozzi et al., "NoC Synthesis Flow for Customized Domain Specific Multi-Processor Systems-on-Chip", IEEE Transactions on Parallel and Distributed Systems, Feb 2005.

[20] M. Dall'Osso et. al, "xpipes: a Latency Insensitive Parameterized Network-on-chip Architecture For Multi-Processor SoCs", pp. 536-539, ICCD 2003.

[21] A. Pinto et al., "Constraint-Driven Communication Synthesis", Proc. DAC 2002.

[22] K. Sekar et al., "FLEXBUS: a high-performance system-on-chip communication architecture with a dynamically configurable topology", Proc. DAC 2005:

[23] M. B. Taylor et al., "Scalar Operand Networks: On-chip Interconnect for ILP in Partitioned Architectures", HPCA 2003.

[24] J. Rabaey et al., "Digital Integrated Circuits", Prentice Hall, 2002.