NoC Topology Synthesis for Supporting Shutdown of Voltage Islands in SoCs

Ciprian Seiculescu[⋆], Srinivasan Murali[§ ⋆], Luca Benini[‡], Giovanni De Micheli[⋆]
[⋆] LSI, EPFL, Lausanne, Switzerland,{ciprian.seiculescu, giovanni.demicheli}@epfl.ch
[§] iNoCs, Lausanne, Switzerland, murali@inocs.com
[‡] DEIS, Univerity of Bologna, Bologna, Italy, luca.benini@unibo.it

ABSTRACT

In many *Systems on Chips (SoCs)*, the cores are clustered in to *voltage islands*. When cores in an island are unused, the entire island can be shutdown to reduce the leakage power consumption. However, today, the interconnect architecture is a bottleneck in allowing the shutdown of the islands. In this paper, we present a synthesis approach to obtain customized application-specific *Networks on Chips (NoCs)* that can support the shutdown of voltage islands. Our results on realistic SoC benchmarks show that the resulting NoC designs only have a negligible overhead in SoC active power consumption (average of 3%) and area (average of 0.5%) to support the shutdown of islands. The shutdown support provided can lead to a significant leakage and hence total power savings.

Categories and Subject Descriptors

B.4.3 [**INPUT/OUTPUT AND DATA COMMUNICATIONS**]: Interconnections (Subsystems)—*topology*

General Terms

Design

Keywords

NoC, voltage islands, shutdown, leakage power, topology

1. INTRODUCTION

Power management is a challenge for modern Systems on Chips (SoCs), as many of them are destined for the embedded market and have to operate with low power consumption. With technology scaling, the leakage power consumption is increasing rapidly as a fraction of the total power consumption. In fact, leakage power can be responsible for 40% or more of the total system power [6].

In order to reduce the leakage power consumption, cores that are not used by an application can be shutdown or placed in sleep mode, while the other cores can be operational. For example, power gating using sleep transistors is a popular way to shutdown cores [6]. To achieve power gating, the sleep transistors are added between the actual ground lines and the circuit ground (also called the virtual ground) [6], which are turned off in the sleep mode to cut-off the leakage path. Due to routing restrictions, separate VDD and ground lines cannot be used for each core. Instead, cores are grouped in to *Voltage Islands (VIs)*, with cores in an island using the same VDD and ground lines [5]-[8]. When all the cores in an island are unused for an application, the entire island can be shutdown. For example, the IBM fabrication processes CU-08, CU-65HP and CU-45HP all support the partitioning of chips into multiple VIs and power gating of the VIs [4].

In today's SoCs, the interconnect architecture is a bottleneck in allowing the shutdown of the islands. There are several approaches presented to synthesize application-specific Networks on Chips (NoCs) [12]-[15]. However, none of them consider the issue of shutdown of VIs. These approaches cannot be directly extended to design NoCs for SoCs with voltage islands. If the NoC is designed using such approaches, either the whole NoC should be placed in a separate VI or the islands cannot be shutdown.

Placing the entire NoC in a separate VI is not a feasible solution. The NoC switches are usually spread across the chip, connecting the different cores. If the entire NoC is in the same island, it is difficult to route the VDD and ground lines for the NoC across the chip. On the other hand, if all the NoC switches are physically clustered and placed in the center of the chip, then long wires are needed to connect all the cores to the NoC island. Thus, the routing congestion would be enormous and the solution is not scalable. Moreover, additional resources for routing the additional voltage and ground lines may not even be available in the design. If the switches are spread across the different VIs and if a VI needs to be shutdown, then packets between cores on the other VIs that use the switches in this VI cannot be transmitted. This will prevent the shutdown of the entire island.

The concept of voltage island should be considered during the NoC topology synthesis phase itself. In this work, we present a synthesis approach to determine the best NoC topology points that are tailored to meet the application performance constraints, minimizing power consumption and supporting the ability to shutdown voltage islands. To the best of our knowledge, this is the first work that addresses custom NoC topology synthesis for supporting the shutdown of voltage islands.

2. RELATED WORK

Power gating of designs has been widely applied in many SoCs [5]. In [5], the authors present the importance of partitioning cores in to voltage islands for power reduction. Several methods have been presented to achieve shutdown of islands [5]-[8]. Any of these methods can be used in conjunction with our topology synthesis process to achieve the actual shutdown of cores.

A description of the NoC paradigm with the related benefits and issues is presented in [1]-[3]. Many works have been presented on synthesizing bus based systems [16]-[18]. In [9]-[11], algorithms for mapping application to regular NoC topologies are presented. In [12]-[15], methods for designing application specific NoCs are described. However, none of these works address the issue of sup-

DAC'09, July 26-31, 2009, San Francisco, California, USA

porting shutdown of voltage islands on the chip.

In [24] an architecture for *Globally Asynchronous Locally Synchronous (GALS)* NoC is presented. In [23] the authors present a physical implementation of multi-synchronous NoC. In [22], the authors present a methodology to partition a NoC into multiple voltage islands. This work is complementary to ours, as we present a methodology to design a custom NoC topology with VIs. In [19], the authors present an approach to design NoCs with voltage islands. However, the designs produced by the method do not support the shutdown of the islands.

In [20], the authors present approaches to route packets even when parts of the NoC have failed. A similar approach can be used for handling NoC components that have been shutdown. However, such methods do not guarantee the availability of paths when elements are shutdown. Moreover, mechanisms for re-routing and re-transmission can have a large area-power overhead on the NoC [21] and are difficult to design and verify.

3. PROBLEM DESCRIPTION

In this section, we describe the architectural features of the NoC and the synthesis problem.

3.1 Architecture Description

An example of the architecture for which our custom NoC synthesis algorithm is designed is presented in Figure 1. The cores of the design are assigned to different VIs, which is given as an input to our method. The cores in a VI have the same operating voltage (same power and ground lines), but could have different operating frequencies. In order to have a scalable solution, we build NoC systems, where cores in a VI are connected to switches in the same VI. We follow an approach similar to the *GALS* approach, where the NoC components in a VI are synchronous and operate at the same frequency. Having a locally synchronous design eases the integration of the NoC with standard back-end placement&routing tools and industrial flows. Moreover, if the different switches in a VI operate at different frequencies, power and latency hungry synchronizers are needed to connect them.

The cores are connected to the NoC switches by means of Network Interfaces (NIs) that convert the protocol of the cores to that of the network. The NIs also perform clock frequency conversion, if the cores are running at different frequencies than the switches in the VI. When a switch in one VI is to be connected to a switch in another VI, we use a bi-synchronous FIFO to connect them together. The FIFO takes care of the voltage and frequency conversion across the islands. The frequency conversion is needed because the switches in the different VIs could be operating at different frequencies. Even if they are operating at the same frequency, the clock tree is usually built separately for each VI. Thus, there may be a clock skew between the two synchronous islands. We use over the cell routing with unpipelined links to connect switches across different VIs, as the wires could be routed on top of other VIs.

3.2 Synthesis Problem

In our synthesis procedure, we generate switches in each VI to connect the cores in the VI. Optionally, our method can explore solutions where a separate NoC VI can be created. We take the availability of power and ground lines for the intermediate VI as an input, and our method will use the intermediate island, only if the resources are available. Our method produces topologies such that a traffic flow across two different VIs can be routed in two ways: (i) the flow can go either directly from a switch in the VI containing the source core to another switch in the VI containing the destination core, or (ii) it can go through a switch which is placed in the intermediate NoC VI, if the VI is available. The switches in the intermediate VI are never shutdown. The method will automatically explore both alternatives and choose the best one for meeting the application constraints.

The objective of our synthesis method is to determine the number of switches needed in each VI, the size of the switches, their operating frequency and routing paths across the switches, such that application constraints are satisfied and VIs can be shutdown, if needed. Our method determines if an intermediate NoC VI needs to be used to connect the switches in the different VIs and if so, the number of switches in the intermediate island, their sizes, frequency of operation, connectivity and paths. The synthesis method can be plugged in our design flow presented in [15] in order to generate fully implementable NoCs

Our method produces several design points that meet the application constraints with different switch counts, with each point having different power and performance values. The designer can then choose the best design point from the trade-off curves obtained.

4. TOPOLOGY SYNTHESIS APPROACH

The synthesis algorithm is explained in detail in this section. From the input specifications, we construct the VI communication graph defined as follows:

DEFINITION 1. *A VI Communication Graph (VCG(*V*,* E*,* isl*)) is a directed graph, each vertex* $v_i \in V$ *represents a core in the VI denoted by* isl *and the directed edge* (v_i, v_j) *representing the communication between the cores* v_i *and* v_j*. The bandwidth of traffic flow from cores* v_i *to* v_j *is represented by* $bw_{i,j}$ *and the latency constraint for the flow is represented by* $lat_{i,j}$*. The weight of the edge* (e_i, e_j)*, defined by* $e_{i,j}$*, is set to a combination of the bandwidth and the latency constraints of the traffic flow from core* v_i *to* v_j*:* $h_{i,j} = \alpha \times bw_{i,j}/max\_bw + (1-\alpha) \times min\_lat/lat_{i,j}$*, where* max_bw *is the maximum bandwidth value over all flows,* min_lat *is the tightest latency constraint over all flows and* α *is a weight parameter.*

The value of the weight parameter α can be set experimentally or obtained as an input from the user, depending on the importance of performance and power consumption objectives.

Algorithm 1 Core-to-switch connectivity

```
1: Determine the frequency at which the NoC will operate in each
   VI and max_sw_size_j, ∀j ∈ [1 ··· N_VI]
2: min_sw_j = |VCG(V, E, j)|/max_sw_size_j, ∀j
3: {Vary number of switches in each VI}
4: for i = 1 to max_{∀j∈1···N_VI} |V_j| do
5:    for j = 1 to N_VF do
6:       if i + min_sw_j < |V_j| then
7:          k = i + min_sw_j
8:       else
9:          k = |V_j|
10:      end if
11:      Perform k min-cut partitions of VCG(V, E, j).
12:   end for
13:   {Vary number of switches in intermediate NoC VI}
14:   for k = 0 to max_{∀j∈1···N_VI} do
15:      Compute least cost paths for inter-switch flows. Choose
         flows in bandwidth order and find the paths.
16:      If paths found for all flows save design point
17:   end for
18: end for
```

The algorithm for topology synthesis is presented in Algorithm 1. In the first step of the algorithm, the frequencies of operation

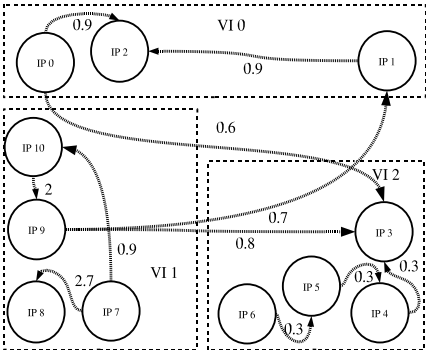

Figure 1: Example Input

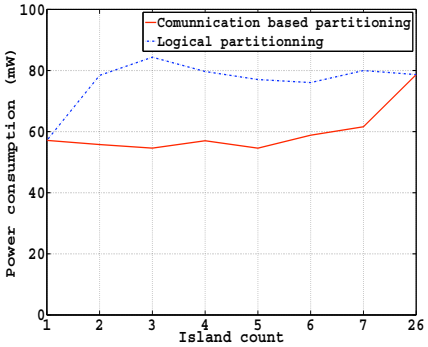

Figure 2: VI count vs. power

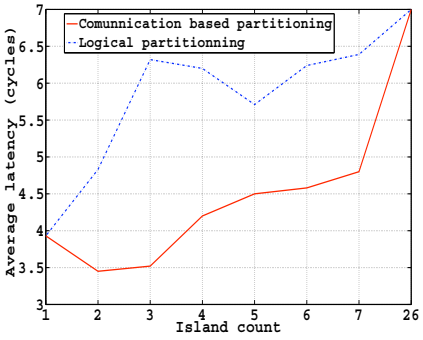

Figure 3: VI count vs. latency

of the switches in each of the islands are determined. In our NoC design, a core is connected to only one switch, through a NI. The links connecting the NI and the switch determine the frequency at which the NoC elements have to run in an island. The bandwidth available on a link is a product of the link data width and the frequency. In our synthesis procedure, without loss of generality, we fix the data width of the NoC links to a user-defined value. Please note that it could be varied in a range and more design points could be explored, which does not affect the algorithm steps. For a fixed data width, the frequency of the switches in an island is determined by the link that has to carry the highest bandwidth from or to a core in the island.

A larger switch will have a longer critical path in the crossbar and therefore will have to operate at a smaller frequency. The frequency at which a switch has to operate determines the maximum size (number of inputs and outputs) of the switch that can be allowed, denoted as $max\_sw\_size_j$. As the switches in the different VIs can operate at different frequencies, the maximum switch size is different for the different VIs. Once the maximum switch sizes are determined, based on the number of cores in each VI, the minimum number of switches required for each island is determined in step 2 of the algorithm. Let N_{VI} denote the total number of VIs in the design. In steps 4 to 10 of the algorithm, the number of switches in each island is varied from the minimum value (computed in step 2) to the maximum number of cores in the island. In step 11, for the current switch count of the VI, that many min-cut partitions of the VCG corresponding to the VI are obtained. Cores in a partition share the same switch. As min-cut partitioning is used, cores that communicate heavily or that have tighter latency constraints would be connected to the same switch, thereby reducing the power consumption and latency.

At this point, the connectivity of the cores with the different NoC switches is obtained. We still need to connect the switches together and find paths for the inter-switch traffic flows. If the switches from a VI are directly connected to the switches on the other VIs, then several switch-to-switch links would be needed. This may lead to large switch sizes, which may lead to violation of the $max\_sw\_size_j$ constraint. By using switches in an intermediate NoC island, the number of switch-to-switch links can be reduced. These switches act as indirect switches, as they are not directly connected to the cores, but only connect other switches. If the design constraints permit the usage of another VI, then we explore the solution space (step 14) with varying number of switches on the intermediate NoC VI.

For each combination of direct and indirect switches, the cost of opening links is calculated and the minimum cost paths are chosen for all the flows (step 15). The traffic flows are ordered based on the bandwidth values and the paths for each flow in the order is computed. The cost of using a link is a linear combination of the power consumption increase in opening a new link or reusing an existing link and the latency constraint of the flow. When opening links, we ensure that the links are either established directly across the switches in the source and destination VIs or to the switches in the intermediate NoC island.

If for all the flows, paths that do not violate the latency constraints are found, then the design point is saved. Finally, for each valid design point, the NoC components are inserted on the floorplan and the wire lengths, wire power and delay are calculated. The time complexity of our algorithm is $O(V^2E^2ln(V))$, however in practice the algorithm runs quite fast as the input graphs typically are not fully connected.

5. EXPERIMENTAL RESULTS

Experiments are performed using the power, area and latency models for the NoC components based on the architecture from [25]. The models are built for 65nm technology node. We extended the library with models for the bi-synchronous voltage and frequency converters.

When the NoC has to be designed to support power gating of islands, there is an additional overhead on the dynamic power consumption of the NoC. To study the impact of the overhead and to see the impact of different core assignment to islands and different number of islands, we consider a case-study on a realistic SoC benchmark. The SoC design is used for mobile communication and multimedia applications. The benchmark has 26 cores, consisting of several processors, DSPs, caches, DMA controller, integrated memory, video decoder engines and a multitude of peripheral I/O ports.

We consider two ways of assigning the cores to different VIs. One way, designated as *logical partitioning*, is based on the functionality of the cores. For example, shared memories are placed in the same VI, as they have the same functionality and therefore are expected to operate at the same frequency and voltage. The island with the shared memories is also expected not to be shutdown, since memories are shared and should be accessible at any time and this is another reason to cluster them in the same VI. Similar reasoning was used to partition all the cores for the case of *logical partitioning*. Another way we considered for partitioning is based on communication and is called *communication based partitioning*. In this case, cores that have high bandwidth communication with one another will be placed in the same VI. Please note that the assignment of cores to the VIs is an input to our synthesis algorithm.

In Figure 2, we show how the dynamic power consumption of the NoC varies when the cores are partitioned in to different number of VIs. The power consumption values comprise the consumption on

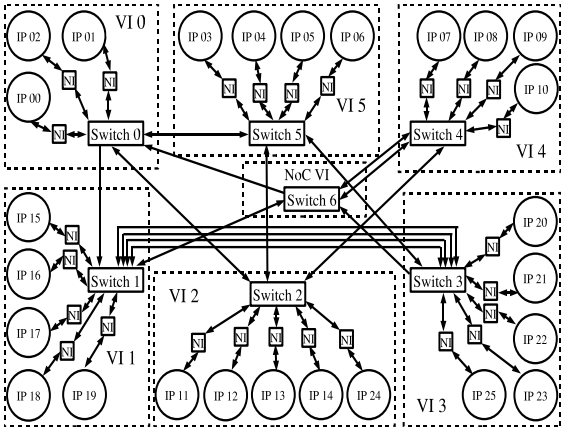

Figure 4: Topology example

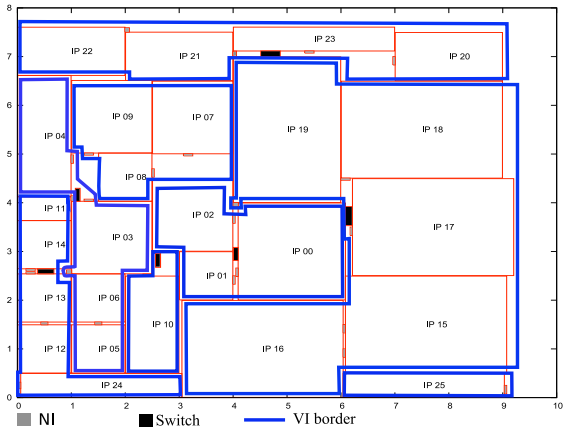

Figure 5: Floorplan example

switches, links and the synchronizers. In the x-axis, the first point (1 island) is actually a design point with all the cores in the same island, which is the reference point. The last point on the graph corresponds to 26 VIs, which is the point when each core is in its own island. It can be seen that in the case of *logical partitioning*, we have to pay a some overhead in NoC dynamic power, as there are more high bandwidth flows that have to go across islands. In the case of the *communication based partitioning*, the NoC consumes less power than the reference point with 1 island, as the NoC can run at a slower frequency in some of the islands. In this case most of the high bandwidth flows are inside an island, so the power overhead is less. In Figure 3, we show the average packet latencies for the different design points. The latency quoted is the number of cycles needed to transfer a single chunk of the packet from the output of the source NI until the input of the destination NI under zero-load conditions. When packets cross the islands, a 4 cycle delay is incurred on the voltage-frequency converters. Thus, with increasing number of islands, the latencies increase. A topology for the 6 VI *logic partitioning* case is shown in Figure 4 and a floorplan example is presented in Figure 5.

We studied the support of voltage islands on a variety of SoC benchmarks. For the different SoC benchmarks, we found that the topologies synthesized to support multiple VIs incur a 3% overhead on the total system's dynamic power. We found that the area overhead is also negligible, with less than 0.5% increase in the total SoC area. In many SoCs, the shutdown of cores can lead to large reduction in leakage power, leading to even 25% or more reduction in overall system power [6]. Thus, compared to the power savings achieved, the penalty incurred in the NoC design is negligible.

The exploration of the design points for all the benchmark took only a few hours on a 2 GHz Linux machine. To be noted that the synthesis process is only run once at design time and therefor the computational time required by the algorithm is negligible.

6. CONCLUSIONS

Stand by and leakage power consumption of the SoC is becoming a large fraction of the total power consumption. Clustering of cores in to voltage islands and shutdown of unused islands is an effective way to reduce the leakage power consumption. The system interconnect has to be designed to ensure proper operation when shutting down voltage islands. In this paper, we presented an approach to synthesize application specific *Networks on Chip (NoC)* interconnects that can effectively support shutdown of islands. The topologies synthesized by our methods have negligible power and area overhead (3% power and 0.5% area, on average), in order to support shutdown. The presented approach also allows the design space exploration of NoCs with different power-performance values that meet the application constraints.

7. ACKNOWLEDGMENT

We would like to acknowledge the financial contribution of CTI under project 10046.2 PFNM-NM and the ARTIST-DESIGN Network of Excellence.

8. REFERENCES

[1] L.Benini and G.De Micheli, "Networks on Chips: A New SoC Paradigm", IEEE Computers, pp. 70-78, Jan. 2002.
[2] P.Guerrier, A.Greiner,"A generic architecture for on-chip packet switched interconnections", Proc. DATE, pp. 250-256, March 2000.
[3] G. De Micheli, L. Benini, "Networks on Chips: Technology and Tools", Morgan Kaufmann, First Edition, July, 2006.
[4] IBM ASIC Solutions, www.ibm.com
[5] D. Lackey et al, "Managing power and performance for System-on-Chip designs using Voltage Islands", ICCAD 2002.
[6] F. Fallah and M. Pedram, "Standby and active leakage current control and minimization in CMOS VLSI circuits.",IEICE Trans. on Electronics, 2005.
[7] A. Sathanur et al., "Multiple power-gating domain (multi-VGND) architecture for improved leakage power reduction", ISLPED 2008.
[8] Q. Ma, E. F. Y. Young, "Voltage Island Driven Floorplanning", ICCAD 2007.
[9] J. Hu et al., 'Exploiting the Routing Flexibility for Energy/Performance Aware Mapping of Regular NoC Architectures', Proc. DATE, March 2003.
[10] S. Murali, G. De Micheli, "SUNMAP: A Tool for Automatic Topology Selection and Generation for NoCs", Proc. DAC 2004.
[11] S. Murali, G. De Micheli, "Bandwidth Constrained Mapping of Cores on to NoC Architectures", Proc. DATE 2004.
[12] A.Pinto et al., "Efficient Synthesis of Networks on Chip", ICCD 2003, pp. 146-150, Oct 2003.
[13] W.H.Ho, T.M.Pinkston, "A Methodology for Designing Efficient On-Chip Interconnects on Well-Behaved Communication Patterns", HPCA, 2003.
[14] J. Xu et al., "A design methodology for application-specific networks-on-chip", ACM TECS, 2006.
[15] S. Murali et al., "Designing Application-Specific Networks on Chips with Floorplan Information", pp. 355-362, ICCAD 2006.
[16] K. Ryu, V. Mooney, "Automated Bus Generation for Multiprocessor SoC Design", Proc. DATE, pp. 282-287, March 2003.
[17] K.Lahiri et al., "Design Space Exploration for Optimizing On-Chip Communication Architectures", IEEE TCAD, pp. 952- 961, June 2004.
[18] S. Pasricha et al., "Floorplan-aware automated synthesis of bus-based communication architectures", Proc. DAC '05.
[19] L. Leung, C. Tsui, "Energy-Aware Synthesis of Networks-on-Chip Implemented with Voltage Islands", DAC 2007.
[20] T. Dumitras et al., "Towards on-chip fault-tolerant communication", ASPDAC 2003.
[21] S. Murali et al., "Analysis of error recovery schemes for Networks on Chips", IEEE D&T, 2005.
[22] U. Y. Ogras et al., ' Voltage-Frequency Island Partitioning for GALS-based Networks-on-Chip ', in Proc. DAC, June 2007.
[23] Miro-Panades, I. et al., "Physical Implementation of the DSPIN Network-on-Chip in the FAUST Architecture", NoC Symposium , 2008.
[24] Bjerregaard, T. et al. "An OCP Compliant Network Adapter for GALS-based SoC Design Using the MANGO Network-on-Chip", Proc. SoC 2005
[25] S. Stergiou et al., "×pipesLite: a Synthesis Oriented Design Library for Networks on Chips", pp. 1188-1193, Proc. DATE 2005.