Chapter 8
Design and Analysis of NoCs for Low-Power 2D and 3D SoCs

Ciprian Seiculescu, Srinivasan Murali, Luca Benini, and Giovanni De Micheli

Abstract Networks-on-Chip (NoC), being a system-level interconnect, can play a major role in achieving low-power SoC designs. In many designs, the cores are grouped in to Voltage Islands (VIs). To reduce the leakage power consumption, an island containing cores that are not used in an application can be shutdown, while the other islands can still be operational. When one or more of the islands are shutdown, the interconnect should allow the communication between islands that are operational. For this, the NoCs has to be designed efficiently to allow shutdown of VIs, thereby reducing the leakage power consumption. In this chapter, we present methods to design NoC topologies that provide such a support for both 2D and 3D ICs. We show how the concept of VIs need to be considered during topology synthesis phase itself. We also make studies to show the benefits of migrating to 3D-stacked chips for realistic applications that have multiple VIs.

8.1 Introduction

Today, portable digital devices are widely available, targeting different application domains. Many of the applications demand high throughput and performance under tight power budgets, as the devices are battery powered. Over the last decade, there has been a lot of focus in tackling this important problem of developing low-power solutions for digital circuits.

The power consumption of a CMOS circuit consists of three different components [8]: (1) dynamic, (2) short-circuit, and (3) static or leakage power consumption. The dynamic or active power consumption is due to the switching of transistors. The short-circuit power consumption is due to the short-circuit current that arises when both the NMOS and PMOS transistors are active, conducting current from supply to ground. The static or leakage power consumption is present even

C. Seiculescu (✉)
LSI, EPFL, Lausanne, Switzerland
e-mail: ciprian.seiculescu@epfl.ch

C. Silvano et al. (eds.), *Low Power Networks-on-Chip*,
DOI 10.1007/978-1-4419-6911-8_8, © Springer Science+Business Media, LLC 2011

when a circuit is not switching. Several factors contribute to the leakage power consumption: sub threshold leakage, gate-induced drain leakage, gate direct tunneling leakage, and reverse-biased junction leakage [12].

With reducing transistor sizes, the leakage power consumption is becoming a significant fraction of the overall circuit power consumption. Moreover, the leakage power consumption also increases tremendously when the operating temperature of the chip increases. In fact, leakage power can be responsible for up to 40% of the total system power [12].

The devices run a variety of applications, and the utilization of the processor, memory, and hardware cores varies across the different applications. For example, a System-on-Chip (SoC) used in a mobile platform supports several applications (or use-cases), such as video display, browsing, and mp3 streaming.To reduce the leakage power consumption, cores that are not used in an application can be shutdown, while the other cores can still be operational. Power gating of designs has been widely applied in many SoCs [20]. If separate voltage lines are routed to every core, then all the cores that are not used for an application can be shutdown. However, this would require separate VDD and ground lines for each core, thereby increasing the routing overhead.

To reduce the overhead, cores are usually grouped into VIs, with cores in an island using the same VDD and ground lines [12, 20, 22, 25, 36, 41]. In [20], the importance of partitioning cores in voltage islands for power reduction is explained in detail. Several methods have been presented to achieve shutdown of islands [12, 20, 22, 25, 36, 41]. A popular technique to shutdown cores is to apply power gating using sleep transistors [12]. In such a method, sleep transistors are inserted between the actual ground lines and the circuit ground (also called the virtual ground) [12], which are turned off in the sleep mode to cut-off the leakage path. When all the cores in an island are unused for an application, the entire island can be shutdown. Many fabrication technologies provide support for VI 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 [19].

NoCs are a scalable solution to connect the cores inside chips [5, 10, 14]. NoCs consist of switches and links and use circuit or packet switching principles to transfer data across the cores. NoCs can play a major role in reducing the power consumption of the entire SoC. The design of the NoC plays an important role in allowing a seamless shutdown of the islands. When one or more of the islands are shutdown, the interconnect should work seamlessly to connect the islands that are operational. For this, the NoCs have to be designed efficiently to allow shutdown of VIs, thereby reducing the leakage power consumption.

The shutdown support presents two distinct challenges for the NoC: the NoC components should be able to handle the multiple frequencies and voltages, and the topology should allow the packets to be routed even when some islands are shutdown. Many NoC architectures support the Globally Asynchronous Locally Synchronous (GALS) paradigm and handle multiple frequencies and voltages. In [7], an architecture for a GALS NoC is presented. In [26], the authors present a physical implementation of multi-synchronous NoC. Architectures for designing

NoCs using GALS paradigm is presented in [3], and architectures for designing NoCs for GALS and DVFS operation are shown in [4]. In [32], the authors present a design methodology for partitioning an NoC into multiple islands and assigning the voltage levels for the islands.

Designing an NoC topology that meets all the application constraints has been addressed by many works. Researchers have targeted designing simple bus-based architectures [21, 33, 35] to regular NoC topologies [17, 27, 28]. Recently, several works have also addressed the issue of designing application-specific custom topologies that are highly optimized for low-power consumption and latency [1, 15, 16, 30, 34, 39, 43, 45]. However, designing a NoC topology that can support partial shutdown of the system has received less attention. In [11], 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 [29] and are difficult to design and verify.

One simple solution that will allow the interconnect to support shutdown of the islands would be to place the entire NoC in a separate VI. However, this is impractical, as the NoC switches could be spread across the floorplan of the chip, thereby physically spreading over multiple VIs. In this case, it is difficult to route the same VDD and ground lines across all the NoC components to keep them in a separate VI. On the other hand, if all the NoC switches are physically placed together in a single (separate) VI, then the core to switch links will become long. This would lead to large wire delay and power consumption. Moreover, this defeats the purpose of using NoC as a scalable interconnect medium.

Designing a NoC topology that not only meets application communication requirements but also allows for shutdown of islands is a challenging task. In this chapter, we present a method to design NoC topologies that allow shutdown of VIs, thereby playing a vital role in achieving a low-power design. We show how the concept of VIs need to be considered during topology synthesis phase itself.

Recently, 3D stacking of silicon layers has emerged as a promising direction for scaling [2, 6, 9, 13, 18, 23, 44]. In 3D stacking, a design is split into multiple silicon layers that are stacked on top of each other. The 3D-stacking technology has several major advantages including smaller footprint on each layer, shorter global wires, and ease of integration of diverse technologies, as each could be designed as a separate layer. A detailed study of the properties and advantages of 3D interconnects is presented in [2] and [42].

In this chapter, we also show how the NoC can be designed for 3D ICs that support VIs as well. We consolidate our works on designing NoCs for supporting VIs, presented in [37], with our works on topology synthesis for 3D ICs, presented in [31, 38]. We study how much performance and power consumption benefit can be achieved for a variety of SoC benchmarks when migrating from a 2D design to a 3D-stacked design. The rest of the chapter is organized as follows: in Sect. 8.2, we present the assumed architecture for VI shutdown and 3D integration; in Sect. 8.4, we formulate the synthesis problem for application-specific

NoC topologies supporting VI shutdown. The algorithm for synthetizing custom NoCs for 2D-Ics with VI is presented in Sect. 8.5 and the extension of the algorithm from 3D is described in Sect. 8.6. Experimental results and analysis of the results are presented in Sect. 8.7.

8.2 Architecture with Voltage Island Support

The design is partitioned into a number of voltage islands. Since for each island a separate voltage line is needed, the use of more islands will lead to difficulty in routing the different voltage lines. On the other hand, with more islands, a finer control of the shutdown mechanism can be achieved, leading to reduction in power consumption. Thus, the number of islands is chosen by the designer by carefully evaluating the tradeoffs. Usually, the designer also iterates on the number of islands and performs an architectural exploration of the design space. For a particular island count, the assignment of cores to the islands is done based on their functionality and also on their position in the floorplan. Since an entire island will be shutdown to reduce power consumption, most of the cores in an island should be active or idle at the same time for a particular application. Also, cores that have heavy communication between them need to be clustered in the same island. This is because the inter-island communication needs to traverse a frequency and voltage crossing interface, which adds to delay and power consumption. Moreover, to keep the routing of voltage lines simple, cores in an island should be located physically close to each other in the floorplan. Thus, assigning cores to islands is a multiconstrained problem and there are several works, such as [25, 32], that address this issue in more detail. In this chapter, we will only cover the complementary problem of designing a hierarchical NoC for supporting the voltage island concept. We will show how the NoC is designed for 2D and 3D ICs and show the performance benefits of 3D integration in such a scenario.

8.2.1 2D SoC Architecture

An example architecture of the 2D system that we consider for the NoC design is presented in Fig. 8.1. The assignment of the cores to different VIs is performed before the NoC design process and is taken as an input. The cores in a VI have the same operating voltage (same power and ground lines) but could have a different operating frequency. Since the inter-island communication passes through voltage and frequency converters, to reduce power consumption and latency, cores in a VI are connected to switches in the same VI. The NoC of an island (switches and links) operates in a synchronous manner, with the components using the same frequency and voltage value. This is compliant with the Globally Asynchronous, Locally Synchronous (GALS) approach, where the NoC in each island is synchronous and the

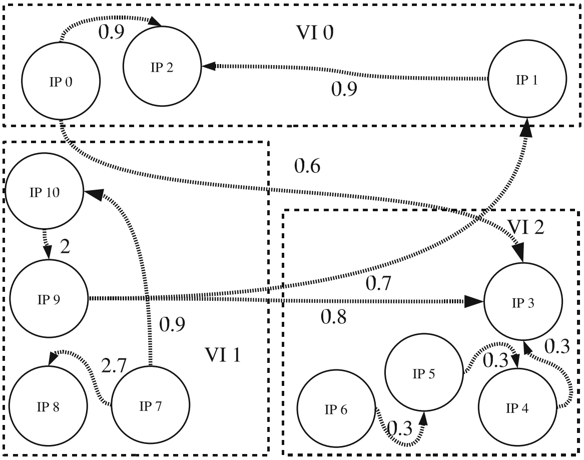


Fig. 8.1 Example input

different islands could have different frequencies and voltages. Having a locally synchronous design also eases the integration of the NoC with standard back-end placement and routing tools and industrial flows.

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. Since the switches in a VI all operate at the same frequency, and the cores can operate at different frequencies, the NIs are also responsible for performing the clock frequency conversion. If switches from different VIs are connected together, then a frequency synchronizer has to be used on the link connecting the two switches. Bi-synchronous FIFO-based frequency converters have been proposed in literature, which can be used for the clock synchronization between switches from different VIs. The FIFOs also take care of the voltage conversion across the islands. Different VIs have different clock trees; therefore, even if they operate at the same frequency, there can be a clock skew between the islands and frequency converters still need to be used. Links connecting switches from different VIs can be routed through other VIs. Hence, for simplicity, we assume that the inter-island links are not pipelined. This restriction can be removed if the pipeline flip-flops are carefully placed in the sending or receiving switch's island.

The shutdown mechanism of an island itself could be performed in many ways. One possible approach for shutdown is the following: when an application does not require cores in a VI, the power manager decides to shutdown the island. The power manager could be either implemented in hardware or could be in the operating system. Before shutting down the island, the manager checks whether all the pending transactions on the NoC into and out of the island are completed, by interrupting or polling all the NIs. If there are no pending transactions and the cores are ready to be shutdown in an island, the power manager grounds the voltage lines to all the cores and network components in the island. In this chapter, we show a general synthesis procedure that is applicable to a system using any shutdown mechanism.

8.3 3D SoC Architecture

We assume a 3D manufacturing process based on the wafer-to-wafer bonding technology. In this, through silicon vias (TSVs) are used for establishing vertical interconnections. A vertical link requires a TSV macro on one of the layers (say the top layer), where the via cuts through the silicon wafer. In the bottom layer, the wires of the link will use a horizontal metal layer to reach the destination. For links that go through more than one layer, TSV macros are required in all the intermediate layers. However, it is important to note that the macros need not be aligned across the layers, as horizontal metal layer can be used to reach the macro at each layer as well. Stacked TSVs are not used, as the alignment of the TSVs would complicate floorplanning. The area of the TSV macros for a particular link width is taken as an input. For the synthesized topologies, our tool automatically places the TSV macros in the intermediate layers and on the corresponding switch ports. Our synthesis process automatically places the TSV macros at different layers for the different vertical interconnects.

An example of the assume architecture is presented in Fig. 8.2. From the bottom layer, the link is first routed horizontally on the metal layer and then vertically. The switch in the top layer has a TSV macro embedded for the port that is connected to this link.

8.4 Design Approach

In this section, we will first show the method for designing the NoC for 2D design to support shutdown of VIs. Then, we show the extensions for designing 3D ICs.

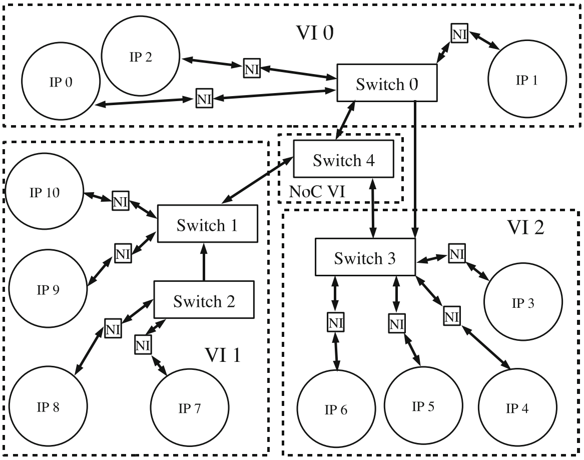


Fig. 8.2 Example architecture

8.4.1 Synthesis Problem Formulation

The synthesis procedure generates switches in each of the VIs. First, the intra-VI communication is tackled, by establishing switches in each VI and connecting the cores of the VI to the switches. Then, the inter VI traffic flows are considered and connections between switches across VIs are established. When connecting switches across VIs, we can either establish a direct connection or use switches in one or more intermediate VIs. For the latter case, we should ensure that the switches in the intermediate islands (apart from the source and destination islands) are not shutdown. A direct connection from a switch in the source island to destination island will lead to lower latency and also usually to lower power consumption. However, when there are too many inter VI flows, the size of switches may increase and the number of inter-layer links and frequency/voltage converters used may also become large. In such a case, using an intermediate VI with switches would be helpful. To ensure proper system operation, this intermediate VI should never be shutdown. The availability of power and ground lines needed to create this intermediate NoC VI is given as input and therefore the usage of the intermediate switches in the NoC VI is optional.

Routes for traffic flows that cross VI bounds are generated by the synthesis algorithm in two ways: (1) 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 (2) 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. If the intermediate NoC VI is allowed, then the method will automatically explore both alternatives and choose the best one for meeting the application constraints.

The objective of the 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. The method we present here also determines whether an intermediate NoC VI needs to be used and if so, the number of switches in the intermediate island, their sizes, frequency of operation, connectivity and paths.

An example NoC design that would be an output of the design approach is presented in Fig. 8.3. As seen, the switches are distributed in the different VIs. The 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.

In order to design the NoC, we should pre-characterize the area, power, and latency models for the individual NoC components (switches, NIs, and bi-synchronous FIFOs). The models can be obtained from synthesis and place and route of the RTL code of the components using commercial tools. The generated library of the NoC components is then used during topology synthesis. As mentioned in the last section, the number of cores and their assignment to VIs are taken as inputs. Optionally, the size and position of the cores are also obtained as inputs. This floorplan information of the cores, if given, will lead to a better estimation

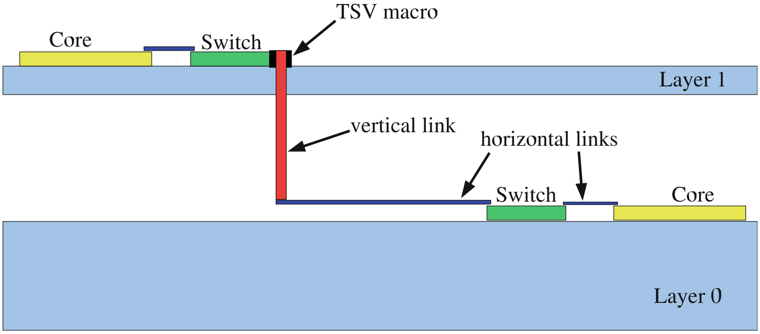


Fig. 8.3 Three-dimensional IC architecture

of the wire power consumption and delays during the synthesis process. Another input is the communication description. In the communication description, for each traffic flow, the source and destination cores are specified and also bandwidth and latency constraints are given.

Based on the inputs and models, we will show methods that synthesize different topology design points. All the topologies generated will comply with the constraints given in the input description files and may have different values for power, average latency, wire length, and switch count. A detailed description of the algorithm is given in the next section. If the size and initial positions of the cores were given as inputs, a floorplan with the NoC will also be generated. The floorplanning routine finds the best location for the NoC components and then inserts the NoC blocks as close as possible to the ideal positions, while minimally affecting the position of the cores given as input.

8.5 Synthesis Algorithm for 2D ICs with VI Shutdown

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

Definition 8.1. 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. communicate with any other core in the same island, edges with low

Algorithm 1 Core-to-switch connectivity

```
Determine the frequency at which the NoC will operate in each VI and $max\_sw\_size_j$, $\forall j \in [1 \cdots N_{VI}]$
$min\_sw_j = |VCG(V, E, j)| / max\_sw\_size_j$, $\forall j$
{Vary number of switches in each VI}
for $i = 1$ to $max_{\forall j \in 1 \cdots N_{VI}} |V_j|$ do
  for $j = 1$ to $N_{VF}$ do
    if $i + min\_sw_j < |V_j|$ then
      $k = i + min\_sw_j$
    else
      $k = |V_j|$
    end if
    Perform $k$ min-cut partitions of $VCG(V, E, j)$.
  end for
  {Vary number of switches in intermediate NoC VI}
  for $k = 0$ to $max_{\forall j \in 1 \cdots N_{VI}}$ do
    Compute least cost paths for inter-switch flows using Check_constraints procedure.
    Choose flows in bandwidth order and find the paths.
    If paths found for all flows save design point
  end for
end for
```

weight (close to 0) are added between the corresponding vertices to all other vertices in the layer. This will allow the partitioning process to still consider such isolated vertices.

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.

In Algorithm 1, we present the steps required to synthesize application-specific NoCs with support for VI shutdown. The first step is to determine the frequency at which the NoC switches have to operate in each island. The minimum required frequency is determined by the highest bandwidth that has to be supported on a link from a NI to a switch. Of course, the NoC in an island can be operated at a higher frequency if the input specifications require, but a lower frequency cannot be supported as one of the cores will not be able to transfer the required bandwidth. The bandwidth available on a link is a product of the link data width and the frequency. For a fixed link width the frequency can be determined. 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.

The frequency at which the switches are operated determines the maximum size (number of inputs and outputs) that the switches can have. Since the critical path of the switch is in the crossbar, there is a direct link between the operating frequency and the size of the switch. In the algorithm, we use $max\_sw\_size_j$ to specify the maximum size that a switch can have in VI_j. Since the frequency in the different VIs is not the same, the maximum switch size will be different in different islands.

Based on the maximum size of the switches and the number of cores in a VI, the minimum number of switches that are required to generate a topology is calculated (step 2). Let N_{VI} denote the total number of VIs in the design.

To better clarify the concepts, we provide examples along with the explanation of the different steps of the algorithm.

Example 8.1. Consider the system depicted in Fig. 8.1. We will describe how the algorithm works for one design point. The design has 11 cores divided into three islands. The first step is to determine the frequency of the NoC in each VI and to calculate the maximum size of the switches in each VI. In this example, IP 7 is generating the maximum traffic in the island VI 1, with a total of 3.6 GB s^{-1} bandwidth. Let us assume that the NoC data width is set to 4 bytes. Thus, the NoC island with IP 7 should run at 900 MHz (obtained by 3.6 GB/4 B). From our NoC libraries at 65 nm, we found that a switch larger than 3 × 3 cannot operate at 900 MHz. Thus, we determine that the maximum switch size for this island is 3 × 3. As the island has four cores, we need at least two switches in the island. The minimum number of switches needed in the other islands can be calculated in a similar manner.

In steps 4–10 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.

Example 8.2. Let us assume that the minimum number of switches computed in step 2 for the example in Fig. 8.3 are 1, 2, 1 for VI 0, VI 1, VI 2. We will generate design points with different switch counts, with each point having one more switches in each VI, until the number of switches is equal to the number of cores in the VI. For this example, we will explore the following points: 1,2,1, 2,3,2, 3,4,3, 3,4,4. As several combinations of switch counts in different VIs are possible, we limit to this simple heuristic.

In step 11, for the current switch count of the VI, 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.

Example 8.3. For the design point 1,2,1, two min-cut partitions of VCG(V,E,1) are obtained. Cores IP 9 and IP 10 communicate more and belong to the same partition. Thus, they would share the same switch. Also, all flows between cores on the same switch will be routed directly through that switch.

Once the connectivity between the cores and the switches in the VIs is established, the algorithm has to find paths and open links for the inter-switch communication flows. Some of these flows also have to cross VI boundaries. For flows that have to cross VI boundaries, a link connecting a switch in the VI with the source core to a switch in the VI containing the destination core has to be found or opened. This can lead to the creation of many new links that can lead to an unacceptable increase in the switch size, violating of the $max\_sw\_size_j$ constraint. If the NoC VI is allowed, then indirect switches in the NoC VI, which are never shutdown, can be used to decrease the size of the switches in the other VIs. These switches act as indirect switches, as they are not directly connected to the cores, but only connect other switches. If the NoC VI is used, then the number of indirect switches is varied in step 14.

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. The different scenarios for setting the link costs are shown in the *Check_constraints* procedure in Algorithm 2. 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. To enforce this constraint, a large cost (*INF*) is assigned to the links that are not allowed. Similarly, when the size of a switch in an island reaches the maximum value, the cost of opening a link from or to that switch is also set to *INF*. This prevents the algorithm in establishing such a link for any traffic flow. Also, when a switch is close to the maximum size (two ports less than the maximum size), a larger value than the usual cost is assigned for opening a new link, denoted by *SOFT_INF*. This is to steer the algorithm to reuse already opened links, if possible. In order to facilitate the use of the indirect switches in the intermediate NoC VI,

Algorithm 2 *Check_constraints*

```
{Check if the link between $switch_i$ or $switch_j$ can be used}
if $island(switch_i) = island(switch_j)$ then
    $link\_allowed =$ TRUE;
else if $island(switch_i) = src\_isl$ and $island(switch_j) = dest\_isl$ then
    $link\_allowed =$TRUE;
else if $switch_i$ or $switch_j$ is in NoC VI then
    $link\_allowed =$TRUE;
else
    $link\_allowed =$FALSE;
end if
$h = island(switch_i)$ and $k = island(switch_j)$
if $size(switch_i) >= max\_sw\_size_h$ or $size(switch_j) >= max\_sw\_size_k$ or
  $link\_allowed =$FALSE then
    $cost_{ij} = INF$
else if $size(switch_i) >= max\_sw\_size_h - 2$ and $switch_j$ is in NoC VI then
    $cost_{ij} = SOFT\_INF/2$
else if $size(switch_j) >= max\_sw\_size_k - 2$ and $switch_i$ is in NoC VI then
    $cost_{ij} = SOFT\_INF/2$
else
    $cost_{ij} = SOFT\_INF$
end if
```

the cost of opening a link between a switch that is close to the maximum size and an indirect switch is set to $SOFT\_INF/2$. Thus, when the size of a switch approaches the maximum value, more connections will be established using the switches in the intermediate NoC VI.

Example 8.4. Let us consider the switch assignment from the previous example. In this example, the highest bandwidth flow that has to be routed first is the one from IP 7 to IP 10. This will result in opening a link between Switch 2 and Switch 1. Now, let us assume that we have to find a path for a flow from IP 9 to IP 3. Because Switch 1 is close to its maximum size and we have other flows to other VIs, the algorithm will use the switch in the intermediate NoC VI. This results in opening a link from Switch 1 to 4 and another from Switch 4 to 3. The topology with the inter-switch links opened is shown in Fig. 8.3.

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 $O(V^2E^2ln(V))$, where V is the set of cores in the design and E is the set of edges representing the communication between the cores. In practice, the algorithm runs quite fast as the input graphs typically are not fully connected.

8.6 Extension for 3D ICs

Extending the algorithm from Sect. 8.5 to generate NoC topologies for 3D-ICs is done by combining the previously presented algorithm with the algorithm for generating custom NoC topologies for 3D-ICs from [31]. In the case of the 3D algorithm from [31], the cores are assigned to layers of the 3D silicon stack. Cores can be connected only to switches in the same layer. The 3D algorithm would explore designs with different number of switches in each layer. The concept of layer is similar to the concept of VI; however, in the original 3D algorithm all layers are assumed to be synchronous.

The extension of the 3D algorithm to support shutdown of VIs as presented in Sect. 8.5 can be done under the assumption that a VI does not span across multiple layers. This assumption makes sense because it is difficult to create a synchronous clock tree that can span on multiple layers. Therefore, even if cores on different layers operate at the same frequency and voltage level, it is very likely that there will be clock skew between them and they would need to be assigned to different VIs.

The algorithm to synthesize NoCs for 3D ICs takes as input both the assignment of cores to the silicon layers of the 3D stack as well as the assignment of the cores to VIs. Also, the maximum number of links that can cross between two adjacent layers has to be given as input. This constraint is used to limit the number of TSVs that are needed to connect components on different layers in order to increase the yield.

8.7 Experimental Results

Experiments reported in this chapter are performed using the power, area, and latency models for the NoC components based on the architecture from [40]. The models are built for 65 nm technology node. We extend the library with models for the bi-synchronous voltage and frequency converters. For reference, the power consumption (with 100% switching activity), area, and maximum operating frequency for some of the components are presented in Table 8.1. The power consumption of a 32-bit link for 1-mm length was found to be 2.72 μW MHz^{-1}. In [24], the authors show that the power consumption of tightly packed TSVs is smaller than that of horizontal interconnect by two orders of magnitude. Therefore, the impact of power consumption and delay of the vertical links is negligible as they are very short as well (15–25 μm). Under zero-load conditions, the switch delay is one cycle, an unpipelined link delay is one cycle and the voltage/frequency converter delay is four cycles (of the slowest clock).

8.7.1 Design of 2D ICs

To support VIs and shutdown of VIs, the NoC will incur an additional overhead due to the use of voltage and frequency converters and because more links need to be opened in order to support all the flows that have to cross VI boundaries. To see how much is the overhead and how it depends on the number of VIs and on the assignment of cores to VIs, we performed experiments on several benchmarks using the 2D algorithm from Sect. 8.5. The first study is performed using a realistic benchmark of a multimedia and wireless communication SoC. The benchmark has 26 cores and its communication graph is presented in Fig. 8.4 [38].

To explore the impact of core to VI assignment on the NoC overhead, we consider two ways of assigning the cores to islands. In one instance, cores that have similar functionality (e.g., shared memories that are never shutdown) or those that are meant to work together (e.g., lower level cache and the processor it services) are assigned to the same VI. The idea is to place cores that are idle at the same time in an VI. We call this assignment *logical partitioning*. This assignment is application oriented and as we will show it will incur higher communication overhead, but it also has more potential for VI shutdown. The other way to partition the cores is based on the communication description. For this assignment of cores to VIs, which we call

Table 8.1 NoC component figures

	Energy (μW MHz^{-1})	Area (μm^2)	Freq (MHz)
Switch 4 × 4	7.2	10,000	803
Switch 5 × 5	8.4	14,000	795
Converter	0.34	1,944	1,000

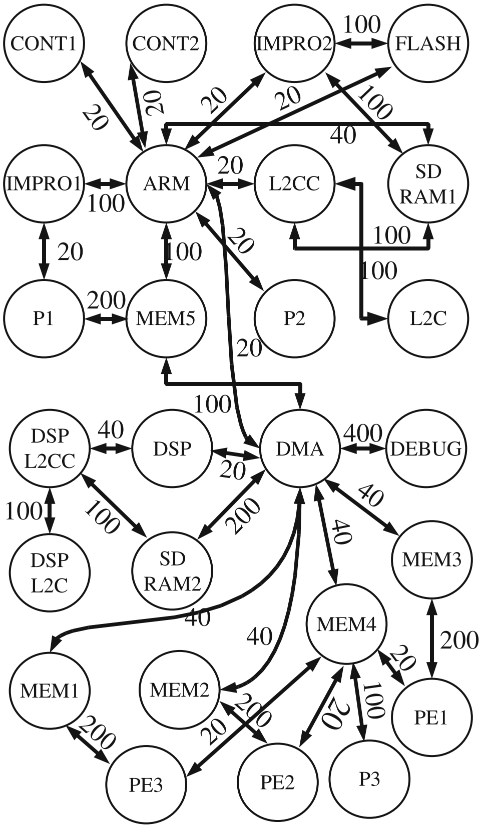


Fig. 8.4 Communication graph

communication-based partitioning cores that have high bandwidth, communication flows are assigned to the same VI. This assignment is communication friendly and will reduce the overhead of the NoC, but the possibilities for VI shutdown are also diminished.

A plot of the dynamic power consumption of the best topology for different numbers of VIs in the design is shown in Fig. 8.5a. There are two values for power point for each VI count. One value corresponds to the case when logical partitioning is used to assign the cores to the considered number of VIs, and the other value corresponds to the assignment by communication-based partitioning. The power consumption values comprise the consumption on switches, links, and the synchronizers. The plot contains the two extremes: the case when there is one VI, which can be used as reference as there is no overhead for the frequency converters. The other extreme is for the 26 VI case when each core is assigned to its own VI. It can be seen that for the *communication-based partitioning*, the overhead on power is not significant until a lot of VIs are used. This is because high bandwidth flows are in the same VI and they do not have to go through the frequency synchronizers. In the

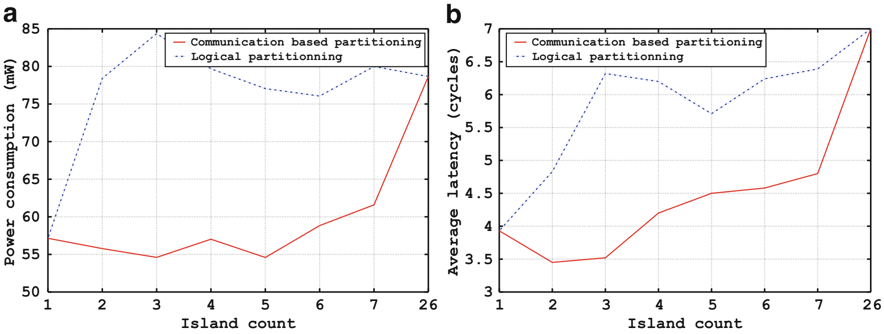


Fig. 8.5 Impact of number of VI on (**a**) power and (**b**) cycles

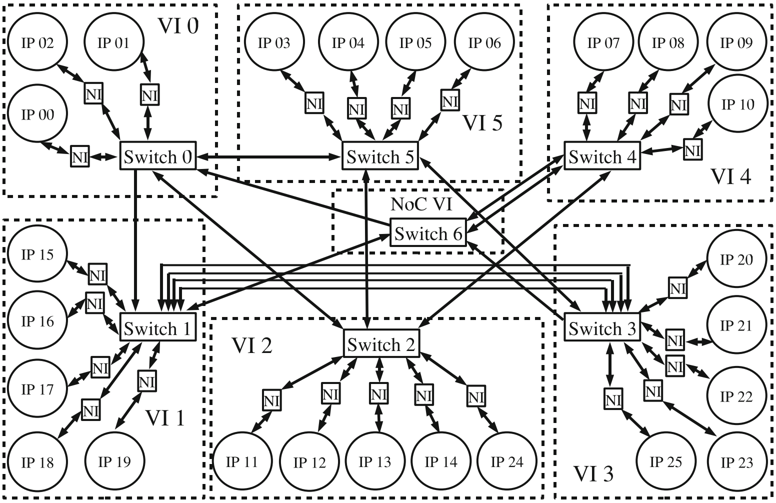


Fig. 8.6 Topology example

case of the *logical partitioning*, we have to pay some overhead in NoC dynamic power, as there are more high bandwidth flows that do go across islands.

We also consider how the average zero load latency is affected by the number of VIs for the two different assignment policies. In Fig. 8.5b, the dependence of the latency on the number of VIs is plotted. The latency value is the average zero load latency of the header flit expressed in cycles. When packets cross the islands, a four-cycle delay is incurred on the voltage-frequency converters. So with the increase in VI, the latency increases as more flows have to go through converters. However, we can see that when the assignment of cores to VIs is done based on communication the average latency does not grow so fast with the number of VIs. This is due to the assignment policy, where more flows are within the same island. A topology for the six VI *logic partitioning* case is shown in Fig. 8.6 and a floorplan example is presented in Fig. 8.7.

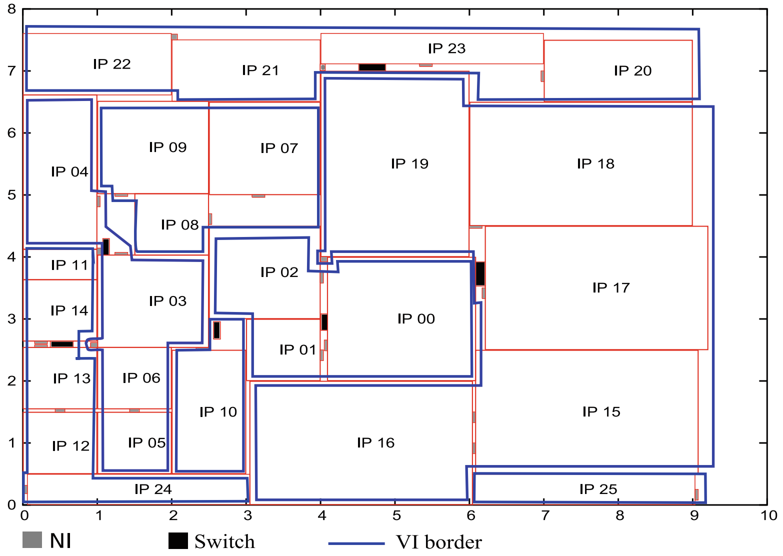
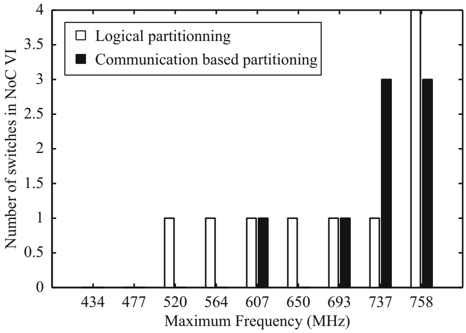


Fig. 8.7 Floorplan example

Fig. 8.8 Impact of frequency on the switch count in the NoC VI



The frequency at which a switch can operate is given by the critical path inside the switch. As the bandwidth requirements of the application start to increase, the required NoC frequencies also increase. Thus, the switches start reaching the maximum allowed sizes to meet the frequency requirements. One way to maintain the size of switches below the threshold is the use of indirect switches in the intermediate NoC VI. However, there is a penalty in using these switches, as a flow going through them has to pass through one more set of voltage-frequency converters. In Fig. 8.8, we show the number of indirect switches used for the best power points when the frequency is varied. We see that when the bandwidth requirements are low, the intermediate island is never used. As the bandwidth requirements start scaling, more and more indirect switches in the intermediate VI are used.

Table 8.2 Comparison on multiple benchmarks

	No VI		Multiple VIs		
	Power (mW)	Latency (cycles)	Power (mW)	Latency (cycles)	VIs
*D*36_4	273.3	4.10	435.5	6.31	6
*D*36_6	295.9	4.17	441.3	7.72	6
*D*36_8	448.5	5.76	561.8	7.71	6
*D*35_*bott*	112.4	5.96	117.82	6.70	6
*D*65_*tvopd*	332.9	3.25	341.64	3.40	8
*D*38_*tvopd*	77.43	3.31	80.12	2.62	4

The presented algorithm automatically explores the entire design space and instantiates the switches in the intermediate island when needed.

In Table 8.2, we present a comparison of power and latency between a design with no VIs and a design with multiple VIs for six benchmarks [38]. Again, we report the dynamic power consumption values of the NoC. We use different number of islands in each benchmark (average of six islands), based on the logical characteristics of the applications. The *D*36_4, *D*36_6, and *D*36_8 set of benchmarks model systems with multiple shared memories on a chip. Each core communicates to 4, 6, and 8 other cores respectively with an average of 2, 3, and 4 communication flows going across the islands. In these cases, the overhead is more as there is a need for many voltage-frequency converters in order to channel all the inter-VI communication. The *D*35_*bott* benchmark has 16 processor cores, 16 private memories, and three shared memories. The processor and the private memory are assigned to the same VI. Thus, there are just the low bandwidth flows going to the shared memories that have to go across VIs. In this case, the power overhead for gating is not significant, and only latency increases since some VIs operate at a lower frequency. In case of the *D*65_*pipe* and the *D*38_*tvopd*, there are 65 and 38 cores respectively, communicating in a pipeline manner and therefore there are fewer links going across VIs, resulting again in a small overhead.

For the different SoC benchmarks, we find that the topologies synthesized to support multiple VIs incur an overhead of 28% increase in the NoC dynamic power consumption. For all the benchmarks, the NoC consumes less than 10% of the total SoC dynamic power. Thus, the dynamic power overhead for supporting multiple VIs in the NoC is less than 3% of the system 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 [12]. Thus, compared to the power savings achieved, the penalty incurred in the NoC design is negligible. Even though the packet latencies are higher when many VIs are used, the presented synthesis approach provides only those design points that meet the latency constraints of the application. Moreover, the synthesis flow allows the designer to perform trade-offs between power, latency, and the number of VIs.

8.7.2 Baseline Comparison of 2D and 3D ICs

Before we showed the effects of VIs on the NoC on 2D-ICs. Now we will show what benefits 3D integration technology can bring from the NoC perspective in designs that use VIs. But first we have to see what is the contribution that 3D technology itself brings to power savings if the circuits can be manufactured in a fully synchronous manner. For this study, we use a media benchmark, *D26_Media*. We consider a three silicon layered IC for the 3D case with the following assignment of cores to layers. The processors and DSPs with their support cores, like the caches and the hardware accelerators, are assigned to the bottom layer. The large shared memories are assigned to the middle layers and the peripherals to the top layer. We use a data width of 32 bits for the NoC links for all the experiments, matching the data width of the cores.

As previously stated, we first consider the case when both the 2D and 3D designs are implemented in a fully synchronous manner. We use this experiment as a baseline to see what is the contribution of 3D technology for power savings and as a reference for the VI overhead for the 2D and the 3D designs. As explained in Sect. 8.5, we determine the minimum required operating frequency based on the highest bandwidth requirement that any core has. For this benchmark , for a single VI, this minimum required frequency is calculated to be 270 MHz. The total NoC power consumption for the best power point for the 2D case is 38.5 mW and for the 3D design is 30.9 mW. Because the total wire length in the 3D case is significantly lower than for the 2D-IC, we obtain a 20% power savings on the NoC. This power saving is only due to the fact that wires are shorter in the 3D-IC (since the benchmark has the same number of cores), and we will show in the next section that under the restrictions of VI assignment, 3D technology can provide higher power savings when compared to 2D.

8.7.3 Comparison for Different Number of Voltage and Frequency Islands

In this section, we present a more detailed analysis and comparisons between 2D and 3D NoC designs when the cores are assigned to different VIs. We made several variations of the *D26_media* benchmark by assigning the cores to different numbers of VIs (from 1 to 7). For this comparison, we only assigned the cores to VIs using the *logical partitioning* policy as described in Sect. 8.7.1. The same VI assignment was used for both the 2D and 3D cases.

The power consumption of the best NoC topologies for different VI counts in the design for 2D-IC is presented in Fig. 8.9a. In the plot, we show the total power consumption and also the break down on the different components. A similar plot showing the power consumption of the best NoC topologies designed for the 3D-IC is presented in Fig. 8.9b. One important thing to note is that, as the number of VIs increases, the operating frequency of certain VIs can be lowered as the cores inside a VI may require less bandwidth than in another. Increasing the number of VIs increases the number of switches in a design, as there has to be at least on switch in each VI. However, when combined with the previous effect that lowers the operating frequency, we can observe that the switch power does not have a significant increase when the number of switches is increased. A similar effect can be observed on the wire power for the 2D case, as the frequency converter is placed closer to the faster switch and the link is operated at the lower frequency. In 3D, however, the switch to switch link power increases marginally with the number of islands, because of the increase in the number of switch to switch links. Power used by the frequency converters grows with the number of VIs for both cases.

In this experiment, we assume a clock skew across the different 3D layers even for a fully synchronous design, thereby leading a minimum of 3 VIs for 3D, with one for each layer. Thus, the total power consumption plotted is the same for one to three VIs in 3D. In Figs. 8.9a, b, we report the actual power values of the different topologies, and in Fig. 8.10 we show the power comparisons of topologies between the 2D and 3D cases. From the latter plot, we can see that the power savings for the case of one VI is only 10%, which is less than what is reported in the previous section. This is because, for the 3D case we actually use 3 VIs, one for each layer as the

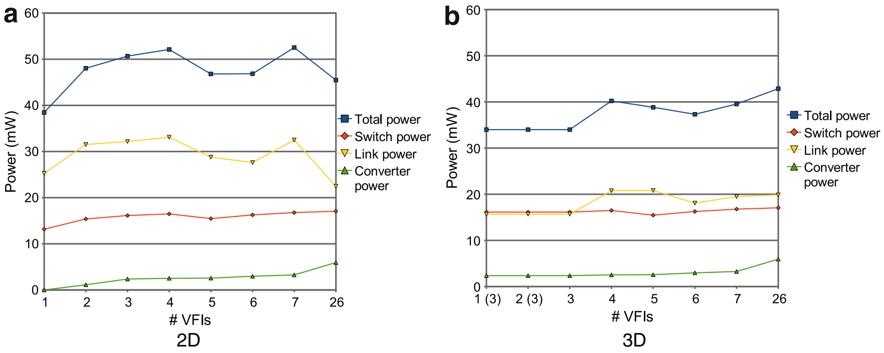


Fig. 8.9 Power for (**a**) 2D and (**b**) 3D designs

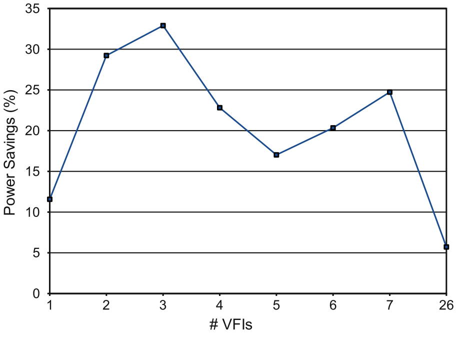


Fig. 8.10 Power savings of 3D over 2D designs

Fig. 8.11 Average zero load latency of 2D and 3D designs

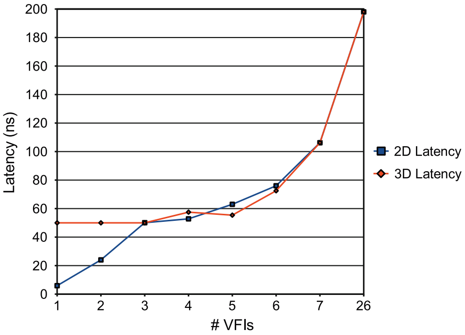
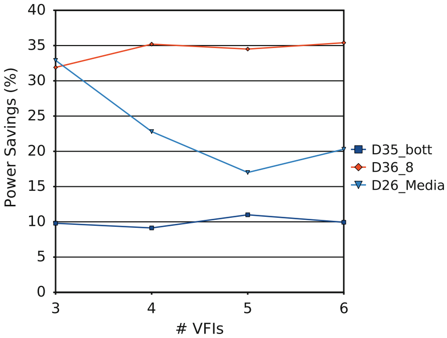


Fig. 8.12 Power savings of 3D over 2D designs for different benchmarks



minimum number. When we compare a solution with 3 VIs in the 2D case as well, we get around 35% interconnect power savings in 3D. However, as the number of VIs increase further, the converter power starts to dominate both 2D and 3D cases. Thus, the power savings achieved by migrating to 3D reduces. Average zero load latency values for topologies designed for different number of VIs for both the 2D and 3D-ICs are shown in Fig. 8.11. Since the links are not automatically pipelined, there is not much difference between 2D and 3D designs. The average latency increases with the number of VIs because more flows have to go through frequency converters, and also because more VIs can be operated at lower frequencies.

To complete the experimental analysis, we also consider two synthetic benchmarks. The *D35_bott* and the *D36_8*, described in Sect. 8.7.1, represent two extremes. The first one has large traffic from processors to their private memories and less traffic to few shared memories. Since the private memories are close to the processors and assigned to the same VIs, the power savings in 3D are small and do not depend too much on the number of VIs, as can be seen in Fig. 8.12. The *D36_8* benchmark is the other extreme with a lot of spread traffic. In this benchmark, all cores have high bandwidth communication to eight other cores. Thus, regardless of the VI assignment, a lot of links will be required in the designs. Hence, we observe high power savings when designing the NoCs for 3D-ICs. Most realistic designs will have communication patterns between these two examples.

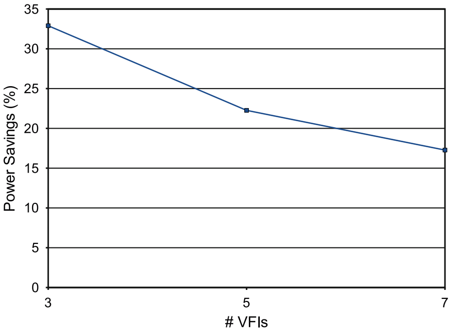


Fig. 8.13 Power savings of 3D over 2D designs for different core areas

So far, we considered the case where VIs are necessary in order to be able to shutdown cores. However, if we look at future technology nodes like 32 nm and beyond, we see that the area of the region where a synchronous clock tree can be built shrinks significantly. Therefore, in large designs, VIs will be needed because it will be very expensive to build a single synchronous clock tree. To capture this effect, we perform an experiment where we increase the size of the cores in the benchmark. As we increase the size of the cores, we also have to increase the number of VIs as it would not be possible to have a single synchronous region. For example, by increasing the size of the cores by 60%, five VIs are needed instead of three if the area of a VI is kept constant. The power consumption difference between the 3D and 2D designs in percentage is shown in Fig. 8.13. As the wires are longer when the benchmark is larger, we obtain more power savings in 3D, when compared to the experiments in the previous subsection.

8.7.4 Analysis of Results

Because the 3D design has three silicon layers and frequency converters or mesochronous synchronizers are needed when a link connects two components on different layers, migration to 3D provides little power saving (11%) with respect to a fully synchronous 2D design with no VIs. However, when more VIs are used in the design (either for functional reasons or due to technological constraints), then NoC generated for 3D SoCs consume much less power than the ones for 2D-IC. In designs with VIs, in order to support shutdown of VIs, more links are needed to route all communication flows that go across VI boundaries; hence, the shorter wire lengths in 3D designs result in considerable power savings (up to 32%). If we increase the number of VIs too much, then the wires become more segmented in 2D as well and get shorter. So, the power saving achieved in 3D NoCs starts to drop. The reduction in the power saving of 3D NoCs with the number of VIs is also due to the fact that the frequency converter power consumption becomes more significant when the number of VIs is increased. There is no significant reduction of the average zero load latency for topologies designed for 3D-ICs. This is because the average zero load delay is dominated by the four cycle delay to cross the frequency converters. The area overhead due to the insertion of TSVs in 3D is negligible, as the TSV macros occupy less than 2% area when compared to the area of the cores.

8.8 Conclusions

Leakage power consumption is becoming a large fraction of the total power consumption in ICs. In order to reduce the leakage power consumption, cores that are not used in an application can be shutdown. For ease of routing signals, cores are grouped into voltage islands and when cores in an entire island are unused, the island can be shutdown. The design of the NoC plays an important role in allowing a seamless shutdown of the islands. The NoC topology should be designed such that even when some islands are shutdown, communication across the different islands that are operational should be possible. In this chapter, we showed how the NoC topology can be designed to achieve this. We applied the methods to design NoCs for both 2D and 3D ICs. Our studies on several benchmarks show a significant reduction in NoC power consumption in migrating to a 3D technology, especially for designs with many voltage islands.

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

References

1. T. Ahonen, D. Signza-Tortosa, H. Bin, J. Nurmi, Topology Optimization for Application Specific Networks on Chip, Proceedings of SLIP, pp. 53–60, Feb 2004
2. K. Banerjee, S. J. Souri, P. Kapur, K. C. Saraswat, 3-D ICs: A Novel Chip Design for Deep-Submicrometer Interconnect Performance and Systems-on-Chip Integration, Proceedings of the IEEE, 89(5):602, 2001
3. E. Beigne, P. Vivet, Design of On-Chip and Off-Chip Interfaces for a GALS NoC Architecture, Proceedings 12th IEEE Intl Symposium on Asynchronous Circuits and Systems (ASYNC 06), IEEE CS Press, 2006, pp. 172–181
4. E. Beigne, F. Clermidy, S. Miermont, P. Vivet, Dynamic Voltage and Frequency Scaling Architecture for Units Integration within a GALS NoC, Proceedings of the Second ACM/IEEE International Symposium on Networks-on-Chip, pp. 129–138, April 07–10, 2008
5. L. Benini, G. De Micheli, Networks on Chips: A New SoC Paradigm, IEEE Computers, pp. 70–78, Jan 2002
6. E. Beyne, The Rise of the 3rd Dimension for System Intergration, Interconnect Technology Conference, 2006 International, pp. 1–5
7. T. Bjerregaard, S. Mahadevan, R. G. Olsen, J. Sparsoe, An OCP Compliant Network Adapter for GALS-based SoC Design Using the MANGO Network-on-Chip, Proceedings 2005 International Symposium on 17-17 Nov 2005 pp. 171–174
8. A. P. Chandrakasan, S. Sheng, R.W. Brodersen, Low-Power CMOS Digital Design, IEEE Journal of Solid-State Circuits, 27(4):473–484, 1992
9. J. Cong, J. Wei, Y. Zhang, A Thermal-Driven Floorplanning Algorithm for 3D ICs, ICCAD, Nov 2004, pp. 306–313

10. G. De Micheli, L. Benini, Networks on Chips: Technology and Tools, Morgan Kaufmann, CA, First Edition, July 2006
11. T. Dumitras, S. Kerner, R. Marculescu, Towards on-chip fault-tolerant communication, ASP-DAC 2003, pp. 225–232
12. F. Fallah and M. Pedram, Standby and Active Leakage Current Control and Minimization in CMOS VLSI Circuits, IEICE Trans. on Electronics, pp. 509–519, Apr 2005
13. B. Goplen and S. Sapatnekar, Thermal Via Placement in 3D ICs, Proceedings of the International Symposium on Physical Design, p. 167, 2005
14. P. Guerrier, A. Greiner, A Generic Architecture for On-Chip Packet Switched Interconnections, Proceedings of the Conference on Design, Automation and Test in Europe, pp. 250–256, March 2000
15. A. Hansson, K. Goossens, A. Radulescu, A Unified Approach to Mapping and Routing on a Combined Guaranteed Service and Best-Effort Network-on-Chip Architectures, Technical Report No: 2005/00340, Philips Research, April 2005
16. W. H. Ho, T. M. Pinkston, A Methodology for Designing Efficient On-Chip Interconnects on Well-Behaved Communication Patterns, HPCA, 2003
17. J. Hu, R. Marculescu, Exploiting the Routing Flexibility for Energy/Performance Aware Mapping of Regular NoC Architectures, Proceedings of the Conference on Design, Automation and Test in Europe. March 2003, pp. 10688–106993
18. W.-L. Hung, G. M. Link, Y. Xie, N. Vijakrishnan, M. J. IRwin: Interconnect and Thermal-Aware Floorplanning for 3D Microprocessors, Proceedings of the ISQED, March 2006, pp. 98–104
19. IBM ASIC Solutions, http://www-03.ibm.com/technology/asic/index.html
20. D. Lackey, P. S. Zuchowski, T. R. Bednar, D. W. Stout, S. W. Gouls, J. M. Cohn, Managing power and performance for System-on-Chip designs using Voltage Islands, Proceedings of the ICCAD 2002, pp. 195–202
21. K. Lahiri, A. Raghunathan, S. Dey, Design Space Exploration for Optimizing On-Chip Communication Architectures, IEEE TCAD, 23(6):952–961, 2004
22. L. Leung, C. Tsui, Energy-Aware Synthesis of Networks-on-Chip Implemented with Voltage Islands, Proceedings of DAC 2007, pp. 128–131
23. S. K. Lim, Physical Design for 3D System on Package, IEEE Design and Test of Computers, 22(6):532–539, 2005
24. I. Loi, F. Angiolini, L. Benini, Supporting Vertical Links for 3D Networks On Chip: Toward an Automated Design and Analysis Flow, Proceedings of Nanonets 2007, pp. 23–27
25. Q. Ma, E. F. Y. Young, Voltage Island Driven Floorplanning, Proceedings of ICCAD 2007, pp. 644–649
26. I. Miro-Panades, et al., Physical Implementation of the DSPIN Network-on-Chip in the FAUST Architecture, Networks-on-Chip, 2008. Second ACM/IEEE International Symposium on 7–10 April 2008 pp. 139–148
27. S. Murali, G. De Micheli, Bandwidth Constrained Mapping of Cores on to NoC Architectures, Proceedings of the Conference on Design, Automation and Test in Europe, 2004, pp. 20896–20902
28. S. Murali, G. De Micheli, SUNMAP: A Tool for Automatic Topology Selection and Generation for NoCs, Proceedings of the DAC 2004, pp. 914–919
29. S. Murali, T. Theocharides, N. VijayKrishnan, M. J. Irwin, L. Benini, G. De Micheli, Analysis of Error Recovery Schemes for Networks-on-Chips, IEEE Design and Test of Computers, 22(5):434–442, Sep–Oct 2005
30. S. Murali, P. Meloni, F. Angiolini, D. Atienza, S. Carta, L. Benini, G. De Micheli, L. Raffo, Designing Application-Specific Networks on Chips with Floorplan Information, ICCAD 2006, pp. 355–362
31. S. Murali, C. Seiculescu, L. Benini, G. De Micheli, Synthesis of Networks on Chips for 3D Systems on Chips. ASPDAC 2009, pp. 242–247
32. U. Y. Ogras, R. Marculescu, P. Choudhary, D. Marculescu, Voltage-Frequency Island Partitioning for GALS-based Networks-on-Chip, Proceedings of DAC, June 2007

33. S. Pasricha, N. Dutt, E. Bozorgzadeh, M. Ben-Romdhane, Floorplan-aware automated synthesis of bus-based communication architectures, Proceedings of DAC, pp. 65–70 June 2005
34. A. Pinto, L. Carloni, A. Sangiovanni-Vincentelli, Constraint-Driven Communication Synthesis, Proceedings of DAC, pp. 783–788, June 2002
35. K. Ryu, V. Mooney, Automated Bus Generation for Multiprocessor SoC Design, Proceedings of the Conference on Design, Automation and Test in Europe, pp. 282–287, March 2003
36. A. Sathanur, L. Benini, A. Macii, E. Macii, M. Poncino, Multiple Power-Gating Domain (multi-VGND) Architecture for Improved Leakage Power Reduction, Proceedings of ISLPED 2008, pp. 51–56
37. C. Seiculescu, S. Murali, L. Benini, and G. De Micheli, NoC Topology Synthesis for Supporting Shutdown of Voltage Islands in SoCs. In Proceedings of the 46th Annual Design Automation Conference (DAC 2009), pp. 822–825, 2009
38. C. Seiculescu, S. Murali, L. Benini, G. De Micheli, SunFloor 3D: A Tool for Networks on Chip Topology Synthesis for 3D Systems on Chip, 2009, pp. 9–14
39. K. Srinivasan, K. S. Chatha, G. Konjevod, An Automated Technique for Topology and Route Generation of Application Specific On-Chip Interconnection Networks, Proceedings of ICCAD 2005, pp. 231–237
40. S. Stergiou, F. Angiolini, S. Carta, L. Raffo, D. Bertozzi, G. De Micheli, × pipesLite: A Synthesis Oriented Design Library for Networks on Chips, Proceedings of the Conference on Design, Automation and Test in Europe 2005, pp. 1188–1193
41. Y-F. Tsai, D. Duarte, N. Vijaykrishnan, M.J. Irwin, Implications of Technology Scaling on Leakage Reduction Techniques, DAC 2003
42. R. Weerasekara, L.-R. Zeng, D. Pamunuwa, H. Tenhunen, Extending Systems-on-Chip to the Third Dimension: Performance, Cost and Technological Tradeoffs, Proceedings of ICCAD, 2007, pp. 212–219
43. J. Xu, W. Wolf, J. Henkel, S. Chakradhar, A Design Methodology for Application-Specific Networks-On-Chip, ACM Transactions on Embedded Computing Systems (TECS), 5(2): 263–280, 2006
44. P. Zhou, Y. Ma, Z. Li, R. P. Dick, L. Shang, H. Zhou, X. Hong, Q. Zhou, 3D-STAF: Scalable Temperature and Leakage Aware Floorplanning for Three-Dimensional Integrated Circuits, ICCAD, Nov 2007, pp. 590–597
45. X. Zhu, S. Malik, A Hierarchical Modeling Framework for On-Chip Communication Architectures, ICCD 2002, pp. 663–671, Nov 2002