Universiteit
Leiden
The Netherlands

On the power efficiency, low latency, and quality of service in network-on-chip

Wang, P.

Citation
Wang, P. (2020, February 12). *On the power efficiency, low latency, and quality of service in network-on-chip*. Retrieved from https://hdl.handle.net/1887/85165

Version:	Publisher's Version
License:	Licence agreement concerning inclusion of doctoral thesis in the Institutional Repository of the University of Leiden
Downloaded from:	https://hdl.handle.net/1887/85165

Note: To cite this publication please use the final published version (if applicable).

Cover Page

Universiteit Leiden

The handle http://hdl.handle.net/1887/85165 holds various files of this Leiden University dissertation.

Author: Wang, P.
Title: On the power efficiency, low latency, and quality of service in network-on-chip
Issue Date: 2020-02-12

Chapter 2

Background

In this chapter, to better understand the contributions of this thesis, we provide some basic information about NoCs, such as network topologies, routing approaches, flow control approaches, and the basic router architecture. In order to confirm and show the major power consumer in a NoC, we also briefly analyze the power consumption on a NoC. Finally, we introduce the conventional power gating technology used in a NoC.

2.1 Network-on-Chip

In the beginning of this chapter, we start with an example of a NoC in a many-core system. As shown in Figure 2.1, this many-core system has 16 processing elements (PEs) connected to a 4×4 2D NoC. These PEs can be of different types, such as processors, digital signal processors (DSPs), peripheral controllers, memory subsystems, etc. Each PE is connected to one network interface (NI) and this NI is connected to a router in the NoC. The routers are arranged in a 2D mesh topology and connected with their neighbor routers.

A PE uses its NI to access the NoC. The NI accepts different messages from the PE and converts the messages into packets that can be transferred over the NoC. Depending on the message type, a message can be converted into one or multiple packets. A packet consists of one or multiple flits. Typically, one packet has one head flit, one or multiple body flits, and one tail flit. Some packets may have only one head-tail flit. The packets carry payload and routing information. The routing information, such as the source PE/router ID, the destination PE/router ID, etc., is stored in the head flit or the head-tail flit.

A NI injects the packets into the NoC. Based on the routing information carried in packets, the routers determine a routing path for the packet, and the packet is trans-

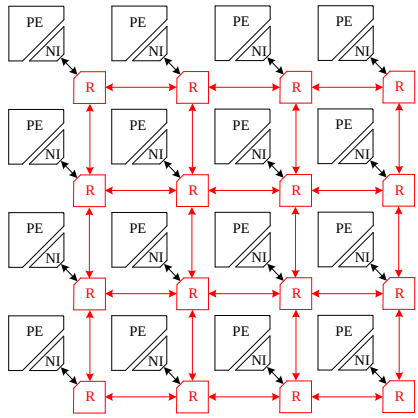


Figure 2.1: An example of a many-core system.

ferred over the routing path to its destination. When the packet reaches it destination router, the destination router ejects this packet into the corresponding NI. The NI collects several packets (if there are multiple packets for a message) and converts these packets into a message that can be processed by the PE.

When designing a NoC, a designer needs to determine the following design factors: the **network topology** (how to connect routers), the **routing approach** (which routing path should be taken to deliver the packets from a source PE/router to a destination PE/router), the **flow control approach** (how to transfer packets between routers), and the **router architecture** (how to implement the routing and the flow control approaches in hardware). Thus, we introduce these design factors in the following subsections.

2.1.1 Network Topologies

The first step in designing a NoC is to choose a proper topology. A given topology arranges the connection of routers and wires in a specific way and has a significant influence on the network performance in terms of network latency and throughput. Currently, most of the real implementations of NoCs utilize simple and regular topologies, such as a ring, a mesh, or a torus.

As shown in Figure 2.2(a), the ring is the simplest topology. The communication on a ring can be easily controlled. However, when the number of routers increases, the network latency of the ring sharply increases, which significantly limits the scalability of the ring topology. Recently, some novel ring-based topologies are proposed. For example, taking the advantage of the simple structure in a ring topology, the hierarchical ring structures [FYNM11, ZML+16] use a ring to increase the communication

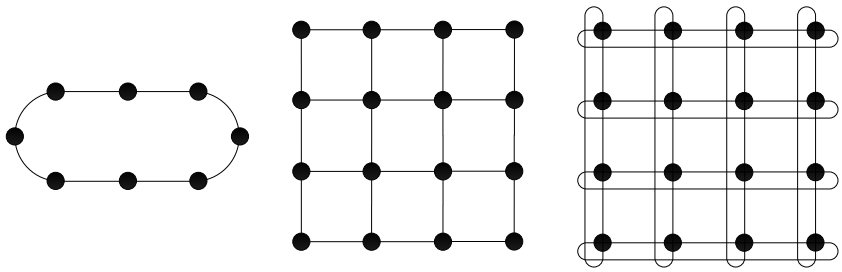


Figure 2.2: Classical network topologies.

bandwidth for the bandwidth-hungry PEs and improve the system performance at a low hardware cost. To improve the poor scalability of a ring, [AABC18, LCL+16] simply use multiple rings to fully connect a large number of routers, which shows excellent performance.

Compared with the ring topology, the mesh topology, shown in Figure 2.2(b), has better scalability and is also widely implemented on real chips. For example, the NoCs in Intel 80-tile [HVS+07], TRIPS [GKS+07], Title-64 [BEA+08], SCORPIO [DCS+14], and SW26010 [FLY+16] adopt such mesh topology. Many novel mesh-based topologys are proposed to reduce the network latency and improve the scalability, such as the flattened butterfly (FBfly) [KBD07], multidrop express channels (MECS) [GHKM09], the concentrated mesh [BD14], the express virtual channel [KPKJ07, KKC+08], the Kilo-NoC [GHKM11], etc. The main drawback of a mesh topology is that most of the traffic workload is concentrated on the routers in the center region of the mesh topology, which results in relatively low throughput of the mesh topology.

The torus topology, shown in Figure 2.2(c), overcomes the aforementioned drawback of the mesh topology by adding long wires to connect the routers on the edges. Thus, the traffic workload can be evenly distributed over the routers. Furthermore, the torus topology has a shorter network diameter than the ring topology and the mesh topology, so the torus has lower network latency. The different length of wires incurs different communication delay between routers, which makes it difficult for controlling the communication between routers. In order to remove the delay gap between the long wires and the short wires in a torus topology, many works utilize the folded torus [DT04, MJW12] topology.

As the mesh topology and the torus topology have better scalability and are widely used in real chips, we implement and evaluate our works using the tours topology in Chapter 3 and the mesh topologies in Chapter 4, Chapter 5, and Chapter 6.

2.1.2 Routing Approaches

A routing approach determines the routing path from a source router to a destination router in a particular topology. Based on the number of the routing paths used form a source router to a destination router, the routing approaches can be classified into:

- Deterministic routing approaches: packets are transferred from a source router to a destination router over exactly the same routing path. In a NoC, there exist multiple feasible routing paths from a source router to a destination router, but deterministic routing approaches always use the same routing path (determined at design-time) from a source router to a destination router, such as X-Y/Y-X dimension-order routing (X-Y/Y-X DoR) [DT01].

- Oblivious routing approaches: packets are transferred from a source router to a destination router over multiple candidate routing paths. These candidate routing paths are randomly selected without considering the state of the network, for example the Valiant's randomized routing [Val82].

- Adaptive routing approaches: based on the network state, the routing path is dynamically selected among multiple candidate routing paths to transfer packets from a source router to a destination router, such as in [FDC+09].

It is easy to implement a deterministic routing approach with a low hardware cost, so the deterministic routing approach is more common in practice, but deterministic routing may cause the problem of an unbalanced traffic workload on different routers, which undermines the resource utilization and degrades the NoC performance. In contrast, the oblivious routing [Val82] and the adaptive routing [FDC+09] are better in balancing the traffic workload on a NoC, but the implementation of oblivious routing approaches and adaptive routing approaches is much more complex than deterministic routing approaches and causes high hardware overhead.

Based on the length of the routing path, the routing approaches can be classified into minimal path routing approaches and non-minimal path routing approaches.

- Minimal path routing approaches: packets are only transferred over the minimal routing paths from a source router to a destination router.

- Non-minimal path routing approaches: packets can be transferred through non-minimal routing paths from a source router to a destination router, such as in [VB81, Val82].

Minimal path routing approaches consume less power because the packets need less hops to reach their destination routers. Thus, most of the NoCs use minimal path routing approaches. However, non-minimal path routing approaches are more promising in achieving better workload balance and realizing fault tolerance in NoCs.

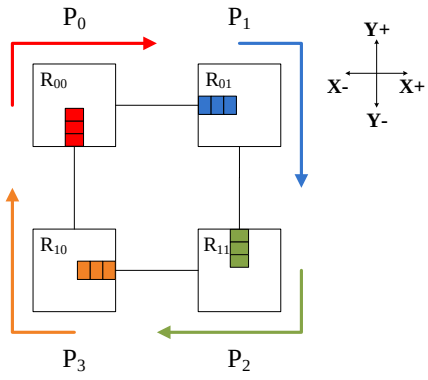


Figure 2.3: Deadlock caused by cyclic dependency of packets.

Avoiding Deadlock

One key role of routing approaches is to avoid deadlock in a NoC. Deadlock occurs when a group of packets are unable to make progress because they are waiting on one another to release resources, usually buffers or virtual channels. Consider the example shown in Figure 2.3. Packet P_0 occupies the buffers in router R_{00} and is going to router R_{01}, but the buffers in router R_{01} are occupied by packet P_1. P_0 has to wait for the buffers in R_{01} to be free. The similar situation happens for packets P_1, P_2, and P_3. As there is a cyclic dependency and all packets are waiting for others to release the buffers, none of the packets can move to the next router and deadlock occurs.

To remove the deadlock in Figure 2.3, the turn model routing algorithm [GN92] is widely used to analyze the dependency between routing paths. According to the turn model routing algorithm, there are eight possible turns in a mesh topology ($X+$ to $Y+$, $X-$ to $Y+$, $X+$ to $Y-$, and so on). These turns can be combined to create two dependent circles (the clockwise circle as shown in Figure 2.3 and the counterclockwise circle), which cause deadlock. By prohibiting some turns to eliminating these dependent circles, the deadlock can be avoid. For example, in the X-Y deterministic order routing (X-Y DoR), packets can be transferred in the $Y+/Y-$ direction only when the packet transmission in $X+/X-$ direction is finished. Thus, the packets cannot turn from $Y+$ to $X+/X-$ or from $Y-$ to $X+/X-$ in a routing path. In this way, all of the dependent circles are eliminated and the deadlock can be avoided.

In Chapters 3, 4, 5, we use the X-Y DoR in the experiments. This is because the X-Y DoR is a deterministic/minimal routing approach, which can be easily implemented and needs less hardware. Furthermore, the X-Y DoR is deadlock free by nature. In

Chapter 6, we implement our approach on a bufferless NoC, which is based on a special adaptive/non-minimal routing approach to guarantee the correctness of the bufferless NoC. This special adaptive/non-minimal routing approach is deadlock free by nature as well.

2.1.3 Flow Control Approaches

After selecting the routing approach, the designers need to choose a flow control approach to transfer packets between routers. The flow control approach determines the packet transmission behavior between routers. Efficient flow control approaches can fully utilize the NoC resources to achieve a low network latency.

Classical Flow Control

The store-and-forward, the virtual cut through, and the wormhole are the most widely used classical flow control approaches in NoCs.

Store-and-forward: A router waits until a packet has been completely received (stored) and then forwards the packet to the downstream router. In addition, the packet being transferred cannot be interrupted by other packets. Store-and-forward incurs high serialization latency [DT04] because routers need to wait for receiving the entire packet.

Virtual cut through : A router can forward a packet as soon as the head flit is received, without waiting for the entire packet to be received, but the packet being transferred cannot be interrupted by other packets. Compared with store-and-forward, virtual cut through removes the serialization latency and is more efficient.

Wormhole: Similar to the virtual cut through flow control approach, wormhole allows routers to transfer packets as soon as the head flit is received. The difference with the virtual cut through is that, in wormhole, the packet transmission can be interrupted by other packets.

Concerning resource allocation, store-and-forward and virtual cut through allocate resources (buffers and wires) at the granularity of packets, while wormhole allocates resources at the much finer granularity of flits. By allocating resources at the granularity of flits, wormhole is beneficial in reducing the amount of required buffers in a router. Furthermore, with more flexible resource allocation, wormhole is able to alleviate the packet blocking and to increase the throughput of a NoC.

Credit-based Flow Control Approaches

The store-and-forward, virtual cut through, and wormhole approaches need buffers in the routers to temporarily store packets. Thus, a means is required to communicate the availability of buffers between an upstream router and a downstream router. Then,

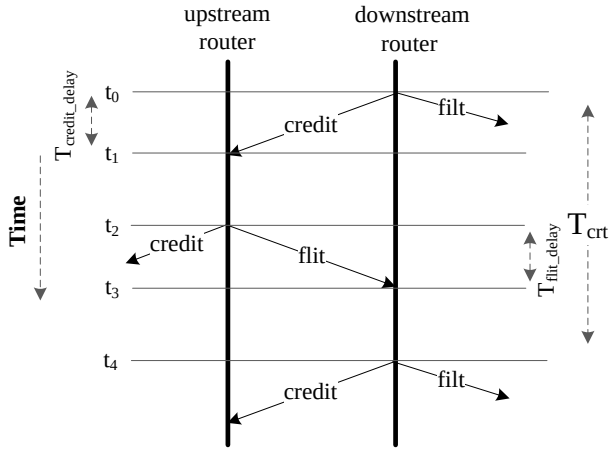


Figure 2.4: A timeline example of a credit-based flow control.

the upstream routers can determine when a buffer is available to hold the next flit (or a packet for store-and-forward and cut-through) to be transmitted. Typically, a credit-based flow control approach is used to monitor the availability of buffers in the routers.

The key idea of the credit-based flow control is that the upstream router keeps a counter of credits, which corresponds to the number of the available buffers in the downstream router. Then, each time the upstream router sends a flit to the downstream router, thereby occupying a buffer in the downstream router, the credit-based flow controller decrements the appropriate credit counter. If the counter reaches zero, all of the downstream buffers are full and no further flits can be forwarded until a buffer becomes available. Once the downstream router forwards a flit and frees the associated buffer, it sends a credit to the upstream router. When the upstream router receives this credit, the upstream router increments the corresponding credit counter to indicate that there is one more free buffer in the downstream router.

An example of the timeline of a credit-based flow control is shown in Figure 2.4. The credit counter in the upstream router is zero and all buffers in the downstream router are occupied. At time t_0, the downstream router sends a flit, thereby freeing a buffer. At the same time, a credit for this buffer is sent to the upstream router. After a short time delay for the credit transmission T_{credit_delay}, at time t_1, the upstream router receives the credit and knows that there is a free buffer in the downstream router. After a short processing delay, at time t_2, the upstream router sends a flit to occupy the free buffer in the downstream router. After a short time delay for the flit transmission T_{flit_delay}, at time t_3, the downstream router receives the flit. After a short period, at time t_4, this flit is forwarded by the downstream router, thereby freeing the buffer again

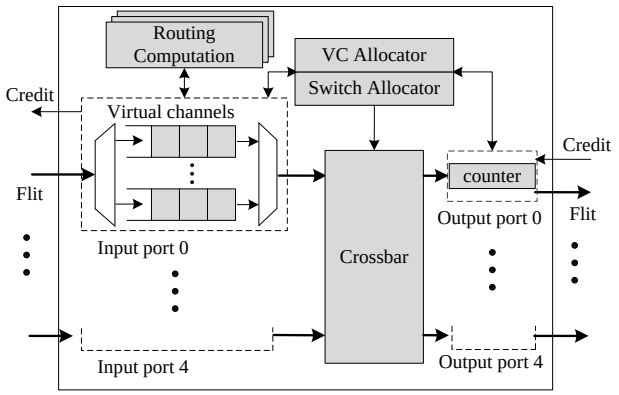


Figure 2.5: Router architecture.

and sending a credit to the upstream router. The time between sending successive credits for the same buffer is the credit round-trip delay T_{crt}. This credit round-trip delay T_{ctr} is the minimal time interval after which the same buffer can be utilized again. In order to reduce the time packets are blocked on waiting for the free buffers, routers need a large number of buffers to hold packets. In this way, the credit round-trip delay can be hidden.

In Chapters 3, 4, 5, we use the wormhole flow control approach and the credit-based flow control approach to control the communication between routers. This is because the wormhole flow control approach needs less buffers than the store-and-forward flow control approach and the virtual cut through flow control approach. Furthermore, the credit-based flow control is the most widely used approach to monitor the availability of buffers in routers. In Chapter 6, we implement the confined-interference communication on a bufferless NoC, in which the flow control is completely different compared to the aforementioned flow control approaches. We will introduce it in Chapter 6.

2.1.4 Router Architecture in NoCs

The router implements the routing approach and the flow control approach in hardware. In this section, we introduce the basic router architecture and the router pipeline.

Router Micro-architecture

As shown in Figure 2.5, a router consists of input ports, routing computation units, a virtual channel allocator unit (VC Allocator), a switch allocator unit, a crossbar, and

output ports.

- **Input ports** are mainly used to temporarily hold packets. In each input port, there are multiple buffer queues (first in first out, FIFO), called virtual channels (VCs). These virtual channels are used for several other purposes, such as avoiding the Head-of-Line blocking [DT01], hiding the credit-round trip delay [DT01], and constructing multiple virtual network, etc. In order to go to the correct VC, each flit of a packet has to carry a VC address, which is used to indicate the VC that the flit should be stored. When a flit of a packet goes into a router, based on the VC address in the flit, the input port stores the flit into the corresponding VC. When a flit of a packet leaves the VC, the input port releases one buffer and sends a credit to inform the upstream router which VC in the downstream router has released one buffer.

- **Routing Computation** unit implements the routing approach for the packets. When there is a head flit of a packet coming, the routing computation unit determines the output port for the packet according to the implemented routing algorithm. In practice, the routing computation unit is separately implemented in each input port to simultaneously compute the output ports for multiple incoming packets.

- **VC Allocator** is a $(N_{upIP} \times N_{upInVC}) \rightarrow (N_{upOP} \times N_{downInVC})$ mapper, where N_{upIP} is the set of input ports in an upstream router; N_{upInVC} is the set of VCs of an input port in the upstream router; N_{upOP} is the set of output ports in the upstream router; $N_{downInVC}$ is the set of VCs of input ports in the downstream routers. When there is a new packet coming, the VC allocator allocates a free VC in the downstream router to the packet. The flits of the packet carry this VC address to the downstream router. When the flits of the packet reach the downstream router, based on this VC address, the downstream router stores the flits of this packet into the corresponding VC. Typically, the VC allocator unit has the most complex hardware logic in a router. The hardware critical path is through the VC allocator. Thus, the complexity of the VC allocator unit limits the operating frequency of a router.

- **Switch Allocator** is a $(N_{upIP} \times N_{upInVC}) \rightarrow N_{upOP}$ mapper. The switch allocator grants the packets to use the crossbar and solves the conflict between multiple packets contending for the same output port. As we use the wormhole flow control, each flit of a packet can go to the downstream router only when the flit gets the grant from the switch allocator.

- **Crossbar** is used to transfer the flits of a packet from an input port to an output port.

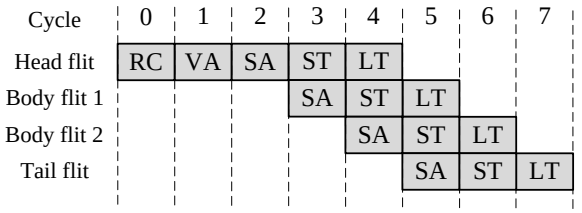


Figure 2.6: Router pipeline.

- **Output ports** contain credit counters, which are used to record the free buffers in each VC in the downstream router. As we have introduced the credit-based flow control in Section 2.1.3, each credit corresponds to a free buffer in the downstream router. When a flit of a packet leaves the router, the credit counter in the output port is decremented to indicate that a buffer space in the downstream router will be occupied. When the output port receives a credit from the downstream router, the output port increments the credit counter, which indicates that there is one buffer released in the downstream router.

In order to transfer packets, a router opperates in several pipeline stages. In a conventional router, there are four pipeline stages: routing computation (RC), virtual-channel allocation (VA), switch allocation (SA), and switch traversal (ST). A packet may experience one more stage called link traversal (LT) in the wires. The pipeline stages in a conventional router are shown in Figure 2.6.

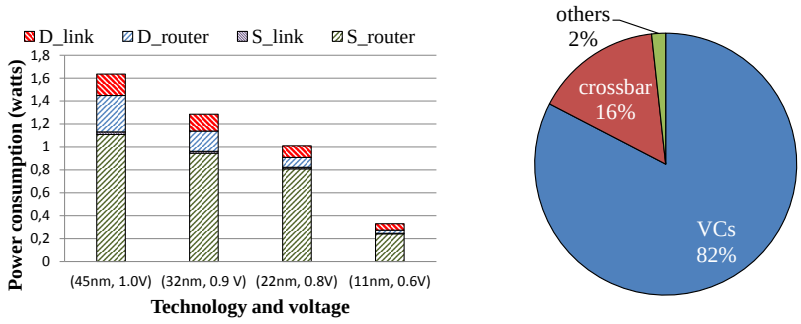
Route computation (RC) stage: The routing computation unit selects a suitable output port according to the routing algorithm. This stage only needs to be performed for the head flit of each packet. The rest of body flits and the tail flit follow the head flit to the same output port.

VC allocation (VA) stage: After the RC stage, the head flit of a packet needs to experience the VA stage to get a free VC in the downstream router. Similar to the RC stage, the VA stage only needs to be performed for head flits, and the rest of the flits in the packet inherit the VC address allocated to the head flit.

Switch allocation (SA) stage: After a packet has been assigned an output port in the current router and a VC in the downstream router, each flit of the packet sequentially requests for permission the switch allocater unit to use the crossbar. The switch allocator unit solves the contention for the same output port between multiple packets.

Switch traversal (ST) stage: After receiving a grant from the switch allocator unit, a flit can traverse the crossbar in the next cycle to reach its destination output port.

Link traversal (LT) stage: Finally, the flit of a packet goes through the links to



(a) Power consumption under different technologies and voltages.

(b) Static power consumption in a router.

Figure 2.7: Power consumption in a 8×8 2D mesh NoC.

the downstream router.

The aforementioned pipeline stages can be hidden or overlapped. For example, based on the Look-ahead routing [Gal97], the RC stage for the downstream router can be executed ahead in the upstream router. When the packet reaches the downstream router, the packet directly goes to the VA stage. Thus, one pipeline stage can be hidden. However, the look-ahead routing is only feasible for the deterministic routing approach, in which the routing path of a packet can be easily determined. Based on the speculative routing [PD01], the VA stage and the SA stage can be performed in parallel. The SA stage for the head flit of a packet is speculative because it depends on the success of the VA stage being performed at the same time. If the VA stage fails, the SA stage will be ignored even if it succeeds.

2.2 Power Consumption Analysis

In order to confirm and show the major power consumer in a NoC, in this section, we briefly analyze the power consumption of each component in a NoC.

We use Dsent [SCK+12] to evaluate the power consumption of a 8×8 2D mesh NoC. Each input port of a router has two 4-flit VCs and the wire bandwidth is 128 Gbits/s. The packet injection rate is 0.1 flits/node/cycle and the flit size is 128 bits. This injection packet rate is much higher than what can be observed in most of the real applications [DMMD09]. This NoC works at the frequency of $1GHz$. We evaluate the power consumption across different technologies and voltages; $(45nm,\ 1.0V)$, $(32nm, 0.9V)$, $(22nm, 0.8V)$, and $(11nm, 0.6V)$.

Figure 2.7(a) shows the power consumption under different pairs of technologies

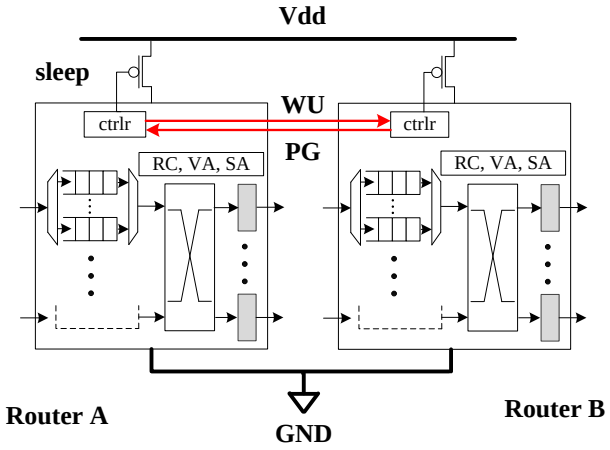


Figure 2.8: Conventional NoC power gating.

and voltages. The total power consumption of a NoC is broken down into four parts: the dynamic power consumption of the link and the routers (D_link and D_router), and the static power consumption of the link and the routers (S_link and S_router). With the downscaling of the technology and the voltage, the total power consumption of a NoC is reduced. However, most of the power consumption is contributed by the static power consumption of routers. For example, the static power consumption of the routers takes 67.78%, 73.55%, 80.17%, and 73.39% of the total power consumption. Thus, in order to reduce the total power consumption in a NoC, the critical point is to reduce the static power consumption of the routers.

In Figure 2.7(b), we show the static power consumption contributed by each component in a router under $(45nm, 1.0V)$. VCs consume the most of the static power. The crossbar and the other components only consume 16% and 2% of the total static power, respectively. Furthermore, in this evaluation, we use two 4-flit VCs in each input port, which is less than what is used in most of the real-word NoCs listed in Table 1.1. So, if the number of VCs further increases, the VCs will consume even more static power. Thus, to reduce the total power consumption, it is crucial to reduce the static power consumption of VCs.

2.3 Conventional Power Gating in A NoC

As we have shown in Section 2.2, the static power consumption takes large portion of the total power consumption. Power gating is an effictive way to reduce the high static power consumption of a NoC. An implementation example of applying conventional power gating on the routers is shown in Figure 2.8. By inserting header transistors

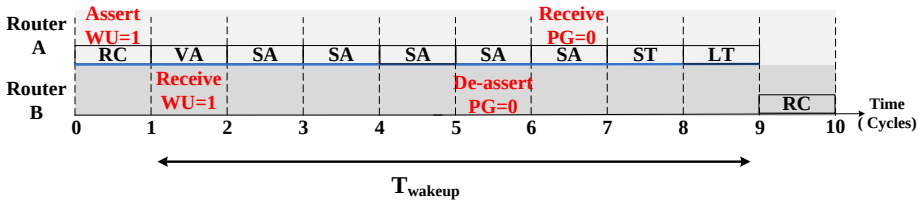


Figure 2.9: Wakeup process.

between the voltage supply and the router, the power controller (the ctrlr unit in Figure 2.8) can cut off the power supply of the router to reduce the power consumption. In order to correctly control the packet transmission, additional handshaking control signals WU (wakeup) and PG (power gating) are added between routers.

When $RouterB$ is idle (there are no flits left in input ports or the crossbar) and the WU signals are clear, the controller in $RouterB$ asserts the sleep signal to cut off the router's power supply (The ctrl unit in routers is always powered-on. Besides, the flow control units that contain the credit counters in output ports are also always powered-on.) and asserts the PG signal to notify its upstream $RouterA$. Once $RouterA$ receives the signal PG, $RouterA$ marks the output port to $RouterB$ as being powered-off. When $RouterA$ needs to send a packet to $RouterB$, $RouterA$ has to assert the WU signal to wake up $RouterB$ and waits for $RouterB$ to be fully charged. Once $RouterB$ is fully charged, the PG signal is de-asserted and $RouterA$ can send the packet to $RouterB$.

An optimized wakeup process [MKWA08, CZPP16] is shown in Figure 2.9. When $RouterA$ executes the RC stage for packets, $RouterA$ determines that there is a packet going to $RouterB$ and asserts the WU signal to wake up $RouterB$. In the following clock cycles, $RouterA$ executes the VA stage and the SA stage, but as $RouterB$ is powered off, the packet has to be blocked in $RouterA$. Once the WU single is received, the ctrlr unit in $RouterB$ clears the sleep signal to charge $RouterB$. After experiencing $T_{wakeup} - MARGIN$ ($MARGIN = 4$ in this example) clock cycles, $RouterB$ de-asserts the PG signal. When $RouterA$ is aware that the PG signal is de-asserted, $RouterA$ allows the packet to go to $RouterB$ and executes the ST stage and the LT stage to transfer the packet. When the packet reaches $RouterB$, $RouterB$ is just fully charged.