PAPER Special Section on Solid-State Circuit Design — Architecture, Circuit, Device and Design Methodology

Hybrid Wired/Wireless On-Chip Network Design for Application-Specific SoC*

Shouyi YIN†, *Member*, Yang HU†, Zhen ZHANG†, Leibo LIU†, *and* Shaojun WEI†, *Nonmembers*

SUMMARY Hybrid wired/wireless on-chip network is a promising communication architecture for multi-/many-core SoC. For application-specific SoC design, it is important to design a dedicated on-chip network architecture according to the application-specific nature. In this paper, we propose a heuristic wireless link allocation algorithm for creating hybrid on-chip network architecture. The algorithm can eliminate the performance bottleneck by replacing multi-hop wired paths by high-bandwidth single-hop long-range wireless links. The simulation results show that the hybrid on-chip network designed by our algorithm improves the performance in terms of both communication delay and energy consumption significantly.

key words: *Network-on-Chip (NoC), on-chip wireless interconnection network, application-specific SoC, multi-core system, design automation*

1. Introduction

The on-chip interconnection network (also known as Network-on-Chip) has been proposed as a promising communication architecture for next generation SoC paradigm [1]. In conventional on-chip network, the on-chip routers are connected by metal wires and the data are transfered in a multi-hop communication manner. Despite the advantages of the on-chip network, the conventional on-chip network faces an important performance limitation that the data transfer between two distant blocks causes high latency and power consumption. Several approaches are proposed to alleviate this problem. In [2] and [3], the long-range links and multi-drop global lines are inserted in a standard mesh network as "shortcuts" to improve the communication performance. However, since these short-cuts are still metal wires, they also suffer from the fundamental physical limitation of material characteristics. To tackle this problem, interconnect innovation with optical, radio frequency (RF) or vertical integration combined with accelerated efforts in design and packaging will deliver the solution [4].

Recently, with the advances of nanotechnology and wireless communication, on-chip wireless communication becomes feasible and can achieve hundreds of GHz to tens of THz of bandwidth [5]–[7]. Based on these advances, several hybrid wired/wireless on-chip network architectures are proposed [9]–[11]. In such hybrid on-chip networks, high-bandwidth single-hop long-range wireless links are employed to transfer data between distants blocks instead of multi-hop wired paths and hence reduce the interconnect delay and energy dissipation. The hybrid wired/wireless on-chip network is a revolutionary on-chip communication infrastructure, which will provide higher flexibility, higher bandwidth, lower power and freed-up wiring when compared to conventional NoC [9].

Compared to a macro wireless data network, a hybrid on-chip network is by far more resource limited. The use of wireless links will cause some additional area cost for implementing the on-chip antenna, modulator and demodulator. To satisfy the overall SoC area requirement, the hybrid on-chip network has to be designed under a strict wireless resource budget (i.e. the number of available wireless links). Moreover, besides the concern about silicon area, since the wireless spectrum is a limited resource, the total bandwidth that can be used in a hybrid on-chip network is also limited inherently. Therefore, in a hybrid on-chip network design, the wireless resources should be used as effectively as possible. Especially in application-specific SoC design, since the traffic characteristics vary significantly across different application, the rare wireless resources have to be judiciously allocated in the on-chip network by taking the traffic and communication pattern into account. However, in most of the previous works [9]–[11], the wireless resources are uniformly distributed in the on-chip network. In [11], besides the uniform hybrid on-chip network architecture, a simple traffic dependent wireless link insertion method is presented. The experimental results show that although the method is fairly intuitive, it can improve the hybrid on-chip network performance notably. This proves the significance of the wireless resource allocation problem for application-specific SoC design.

In this paper, we address the design of hybrid wired/wireless on-chip network, especially the wireless resource allocation problem, for application-specific SoC design. We define the hybrid wired/wireless on-chip network as a combination of conventional on-chip network and some long-range wireless links between distant cores in the chip. Then the key problem is the placement of the wireless links between a particular pair of source and destination cores, which will eventually result in performance gains. Although the problem of bypass channels insertion is well investigated in wired networks, the problem of wireless links allocation still has some unique features that should be tackled. First, in wired network, the distance (or hop-count) between IP cores is usually used as metric to measure the effect of bypass channel insertion. Reducing distance between IP cores

Manuscript received August 11, 2011.
Manuscript revised November 4, 2011.

†The authors are with the Institute of Microelectronics, Tsinghua University, China.

*This work was supported by National Natural Science Foundation of China (No. 60803018).
DOI: 10.1587/transele.E95.C.495

is always the goal of inserting wired bypass channel. However, the wireless links always provide higher bandwidth than wired links, which results in a heterogeneous bandwidth on-chip network. Therefore we cannot simply use distance to measure the effect of wireless link insertion. Otherwise it may cause improper use of wireless links. Some communication performance based methods are needed for inserting wireless links. Second, the wireless link is a lossy channel compared to wired link. To guarantee the reliable data transmission, a retransmission scheme is required in hybrid on-chip network. The retransmission data causes extra communication load in on-chip network. It must be taken into consideration so that we can get accurate performance estimation and allocate wireless links optimally. Third, in most wired bypass channel insertion algorithm, the data flow contention at router is not considered, which impacts the communication performance greatly. Therefore to allocate wireless links efficiently, we must build an analytical model of data flow contention.

To resolve the problem of wireless links allocation, we build an analytical communication performance model for hybrid on-chip network based on the theory of *Network Calculus*. Based on this model, we propose an efficient algorithm that optimizes the placement of wireless links in the on-chip network while matching the communication characteristics of the target application. More precisely, given the total available wireless links, the traffic pattern between different cores, and other relevant architectural parameters (e.g., router processing characteristics and routing algorithm), our algorithm automatically decides the positions where the wireless links are placed. The proposed algorithm is applicable to any topology of on-chip network.

The remaining part of this paper is organized as follows: In Sect. 2, the architecture of hybrid wired/wireless on-chip network is described. The problem formulation of wireless resource allocation is presented in Sect. 3. The resolving methods and optimization algorithms are proposed in Sect. 4. In Sect. 5, we discuss the extension of the proposed algorithm to worst case delay guarantee. Simulation results are described in Sect. 6. Finally, we conclude our paper in Sect. 7.

2. The Architecture of Hybrid Wired/Wireless On-Chip Network

2.1 On-Chip Network Architecture

In a hybrid wired/wireless on-chip network, there are two kinds of communication channels, conventional wired links and long-distance high-bandwidth wireless links. The neighboring IP cores are connected by the wired links and the wireless links are used to connect the distant cores. These on-chip wireless links enable on-hop data transfer between distance cores, which reduce multi-hop long-distance wired communications. Therefore, a hybrid wired/wireless on-chip network can also be deemed as a combination of conventional wired on-chip network with an arbitrary topology and some long-range wireless shortcuts.

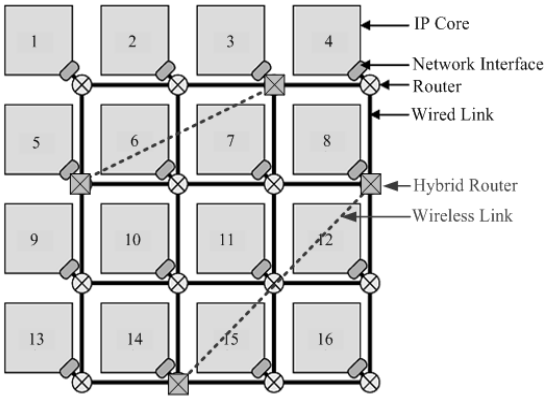


Fig. 1 A hybrid wired/wireless on-chip network.

Figure 1 shows an example of mesh-based hybrid on-chip network. Basically the cores are interconnected by a 2-D mesh wired on-chip network. And there are two long-distance wireless links: one connects Router 3 and 5, the other connects Router 8 and 14. The routers with wireless links (also called hybrid routers) are equipped with wireless communication units that transmit and receive data packets over the wireless channels.

By the state-of-the-art on-chip wireless communication technology, tens of different frequency channels can be created over a single chip. Thus these tens of different frequencies can be assigned to multiple wireless links in the hybrid on-chip network in such a way that single frequency channel is used only once to avoid signal interference. This enables concurrent data transmission over multiple wireless links on the chip. For example, in [11], 24 different frequency wireless links are used in a scalable hybrid on-chip network and each link provides 10 Gbps bandwidth. The on-chip wireless link is also a lossy channel. Within the typical communication range of multi-core SoC ($\sim$1.5 cm), the bit-error rate (BER) of wireless link is less than 10^{-9} [10]. It is far higher than that of RC wires ($\sim 10^{-14}$). Hence, the data loss has to be managed in on-chip hybrid routers.

In application-specific SoC, the traffic characteristics in the on-chip network vary significantly across different applications. We assume that the application-specific nature enables us to characterize traffic with sufficient accuracy. We model the on-chip traffic by the *arrival curve* in *network calculus* [12]. And we also model the packet processing of router by the *service curve*.

2.2 Hybrid Router Microarchitecture

There are two kinds of routers in hybrid on-chip network, basic router and hybrid router. The basic router has the same microarchitecture as conventional on-chip router. And the hybrid router is the combination of a basic router and a wireless communication unit. Figure 2 shows the microarchitecture of a hybrid router in 2-D Mesh based hybrid on-chip network. It is composed of five bi-directional ports named

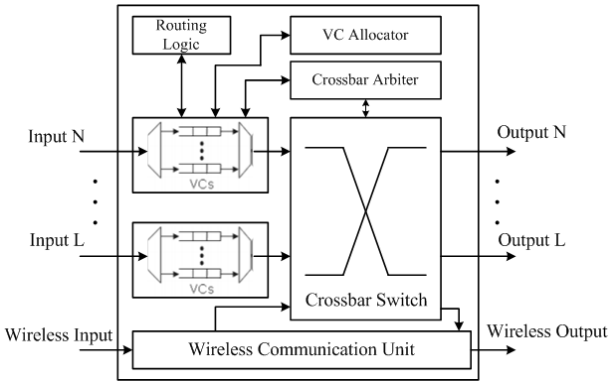


Fig. 2 Hybrid wired/wireless on-chip router microarchitecture.

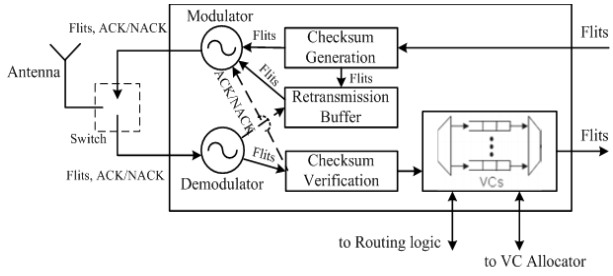


Fig. 3 The block diagram of the wireless communication unit.

as North (N), South (S), East (E), West (W) and Local (L), crossbar switch, control logic and a wireless communication unit. The major control logic components include a routing logic, a VC (Virtual Channel) allocator and a switch arbiter. The wireless communication unit is connected to the crossbar switch so that the flits from wired ports can be forwarded to wireless link and vice versa. But the flits from wireless input is prohibited to be forwarded to wireless output. The algorithm of routing logic is described in Sect. 2.3.

The block diagram of the Wireless Communication Unit (WCU) is shown in Fig. 3. It is composed of modulator, demodulator, checksum generation logic, checksum verification logic, input buffer and retransmission buffer. We use a FDM-like technology in the hybrid on-chip network; several frequencies are assigned to multiple wireless links and single frequency is used only once. Therefore the WCU does not need media access control. The antenna is controlled by a switch to switch between the transmitting and receiving modes. Thus WCU provides a half-duplex wireless channel. The WCU is connected to routing logic and VC allocator for packets routing and switching. To manage the data loss of wireless channel, the error detection codes (parity or cyclic redundancy check codes) are added into the data flits. In the process of communication, the WCU of sender accepts data flits from crossbar switch and after checksum generation and modulation, transmits the data stream through the antenna. At the same time, the sender stores the transmitted data flits into Retransmission Buffer. And the WCU of receiver picks up the signal using the on-chip antenna, and then demodulates, verifies the checksum. If the checksum is right, the flit enters the input buffer and then routes to the output port. Otherwise, an NACK signal will be sent back to the sender. When the sender receives an NACK signal, it will retransmit the corresponding flit from the Retransmission Buffer.

2.3 Routing Mechanism

The key point of routing in the hybrid on-chip network is determining whether to use the wireless link in current hybrid router or not. Since the wireless long-range link always provides much higher bandwidth than the wired link, we cannot just use hop-count as the routing metric. Assuming the current router i is connected to router k by the wireless link and the destination is router j, we need to determine whether use wireless link (i, k) in the path to j. From the performance point of view, if wireless link produces a shorter communication delay, it should be used in current router. Therefore we compare the transmission delay from router i to router j with the wireless link and without the wireless link. If the wireless link (i, k) is used, the transmission delay is

$$M(i, j) = \frac{1}{B_{wl}} + \frac{HC(k, j)}{B} \tag{1}$$

where B_{wl} and B are the bandwidth of wireless link and wired link respectively, and $HC(k, j)$ is the hop-count from router k to router j. Otherwise the transmission delay is $HC(i, j)/B$. If $M(i, j) < HC(i, j)/B$, we determine to use the wireless link in router i. Since this method only uses the hop-count information, it is applicable to any topology and scalable enough. Therefore we can extend the traditional routing algorithm (e.g. *XY* routing, Odd-Even routing, etc.) with the above mechanism for an arbitrary topology hybrid on-chip network. It should be noted that since the inserted wireless link may cause some irregular direction links, such as NE-SW direction link in mesh network, the deadlock should be concerned in hybrid path.

For example, in a 2-D Mesh based hybrid on-chip network, we use *XY* routing as default routing algorithm. Since the wireless links may cause NE-SW and NW-SE direction links, we use South-East routing [2] for the hybrid routers to guarantee a deadlock-free path. The overall routing algorithm is shown in Fig. 4. The algorithm first checks whether there exists a wireless link in the current router i. If there is no such link, the *XY* routing algorithm is used. Otherwise, the algorithm evaluates the transmission delay to destination j when using the wireless link. If the wireless link produces a shorter delay, the wireless link is selected and South-East routing algorithm is used to guarantee freedom from deadlock. Otherwise, the *XY* routing algorithm is used.

2.4 Implementation Cost of Hybrid On-Chip Network

The use of wireless links in on-chip network causes some implementation cost. First, the regularity of the conventional on-chip network is altered. In conventional network, each router has the identical architecture and silicon area;

and each port has the same bandwidth. In contrast, the hybrid router has more complex architecture than basic router and the wireless link can provide higher bandwidth. Therefore more effort is needed in placement and routing to handle the two kinds of heterogeneous routers. Moreover, to route the packets efficiently in heterogeneous bandwidth network, the routing algorithm cannot just use hop-count as routing metric. The bandwidth difference should be taken into account and link (or path) quality based routing algorithm (e.g. the algorithm proposed in Sect. 2.3) is necessary. In our approach, we limit the maximum number of wireless links which can be added to the conventional on-chip network. It will control the irregularity of hybrid on-chip network in some extent.

Second, the implementation of wireless links causes silicon area overhead. The hybrid router needs an additional WCU and each WCU requires antenna, modulator, demodulator, checksum logic and retransmission buffer. The on-chip antenna always consumes the dominant portion of the total area. Table 1 summarizes the area overhead of implementing one wireless link by different wireless technologies [8]–[11]. We can see the area overhead scales down with the operating frequency increasing. Since the proposed hybrid on-chip network employs FDM-like wireless technology at sub-THz or THz frequency, the area overhead of implementing one wireless link should be around $200\,\mu m^2$. However, considering a SoC spreading typically over a die area of $10\,mm \times 10\,mm$, the area overhead of a wireless link is very tiny, while enjoying significant performance improvements.

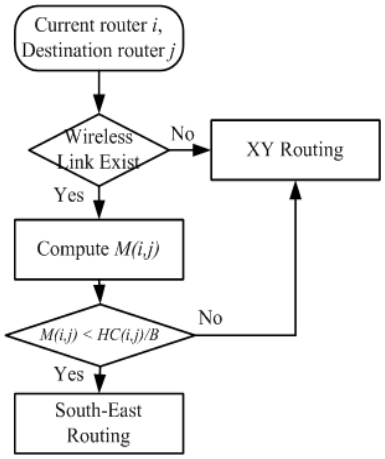


Fig. 4 Routing algorithm for 2-D mesh based hybrid on-chip network.

Table 1 Total area overhead of implementing one wireless link.

Wireless Technology	Frequency (Hz)	Antenna Type	Area Overhead
MM-Wave [8]	60 G	Metal ZigZag	$2.24\,mm^2$
UWB [9]	90 G	Metal	$\sim 0.1\,mm^2$
FDMA [10]	300 G	Metal	$\sim 200\,\mu m^2$
FDM [11]	1T	Carbon Nanotube	$151\,\mu m^2$

3. Problem Formulation of Wireless Link Allocation in Hybrid On-Chip Network

In the hybrid wired/wireless on-chip network, the wireless links are the powerful and rare resources. They are significantly more power and delay efficient compared to their wired counterparts. Replacing multi-hop wired paths in an on-chip network by high-bandwidth single-hop long-distance wireless links results in both reduction of interconnect delay and energy dissipation. On the other hand, the amount of wireless links which can be used in a chip is very limited due to both the silicon area concern and wireless spectrum limitation. Therefore how to effectively use these resources to improve the on-chip communication performance is the essential problem for hybrid on-chip network design.

As mentioned above, the overall interconnect infrastructure of the hybrid on-chip network is the combination of wired conventional on-chip network and some long-distance wireless shortcuts. For the hybrid on-chip network design, the procedure can be split to two step: firstly a conventional wired on-chip network with an arbitrary topology is used to provide the underlying interconnection of the cores; secondly some wireless links are allocated into the wired on-chip network to connect the distant cores [11]. The wireless link allocation aims to find the particular pairs of source and destination IP cores to place wireless links, which will result in performance gains as much as possible.

In this paper, the traffic pattern of an application-specific SoC is modeled by data flows between source and destination cores. Each flow is abstracted by an *arrival curve*. Each router is abstracted by a *service curve*. The end-to-end packet delay is used as the metric for communication performance.

For convenience, we summarize the basic parameters in Table 2. With these notations, the problem of wireless link allocation for performance optimization under total wireless links budget can be formulated as follows:

Given:

- Total wireless links budget N_{wl};
- Application communication flows F and *arrive curve* α_f of each flow;
- *Service curve* β of each router;
- Routing algorithm $\mathcal{R}$;

Table 2 Parameter notation.

Param.	Description
N_{wl}	The total number of available wireless links
$V = \{v\}$	The set of routers in network
$F = \{f\}$	The set of traffic flows in network
$\mathcal{N}$	The conventional wired on-chip network, e.g. $m \times n$ mesh
$\mathcal{R}$	The routing scheme, e.g. XY routing
$\mathcal{HN}$	The hybrid wired/wireless on-chip network
D	The average packet delay
F_v	The set of traffic flows traversing router v
V_f	The set of routers traversed by flow f

• The underlying wired on-chip network $\mathcal{N}$;
Determine:

The particular pairs of routers (v_x, v_y) to place N_{wl} wireless links that can minimize the average packet delay D and *at most one* wireless link is added per router.

4. Wireless Link Allocation Algorithm for Hybrid On-Chip Network

In this section, we present a novel wireless link allocation algorithm that iteratively places the wireless links into the performance bottleneck area of the on-chip network until the specified value of the wireless link budget is reached.

4.1 Hybrid Path Performance Analysis

To allocate the wireless links effectively, the key is to detect the performance bottleneck in the on-chip network. More specifically, given the traffic characteristics and network configuration, the algorithm should be able to identify the most congested path in the network. Hence we need to build an analytical model for a hybrid wired/wireless path. To solve this problem, we resort to the theory of *Network Calculus*.

We use a typical hybrid wired/wireless path shown in Fig. 5 to illustrate the analysis procedure. The path consists of three wired links and one wireless link. And there are two flows: f_1 traverses all five routers and f_2 traverses router R_1 and R_2. Since the wireless link is a lossy channel, the packets retransmission will cause some extra data flows. According to the theory of *Stochastic Network Calculus* [13], the packets loss process can be modeled by a *scaling function* with *scaling curve*. Then the lost packets are modeled as a scaled version of the output flow of the wireless link. Assuming the arrival curve at the sender of wireless link is $\alpha(t) = \gamma_{r,b}(t) = rt + b$, the service curve is $\beta(t) = \beta_{R,T}(t) = R(t - T)^+$ and the scaling curve is $\bar{S}^{\bar{\varepsilon}}(x) = Cx + B$, the arrival curve of the one-retransmission flow is

$$\alpha^{(1)} = \gamma_{Cr,b_\infty}, \text{ where}$$
$$b_\infty = (R - Cr)\frac{RT + Cb + B}{R - 2Cr} - RT \tag{2}$$

The arrival curve of multiple retransmissions can also be calculated similarly [13]. Therefore the hybrid path can be transformed to an equivalent form shown in Fig. 6. For simplicity, only one-transmission flow $f_1^{(1)}$ over the wireless link is shown. Thus the problem is transformed to the performance evaluation of several contention flows over the path.

Considering a general case shown in Fig. 7(a), n flows with arrival curves $(\alpha_1, \alpha_2, ..., \alpha_n)$ traverse a router with service curve β. In [14], the case of $n = 2$ has been discussed. Here we extend it to general case. Considering f_n is the contention flow of the other $n - 1$ flows, according to [14], the equivalent service curve for the flow-set $(f_1, f_2, ..., f_{n-1})$ can be computed as follows:

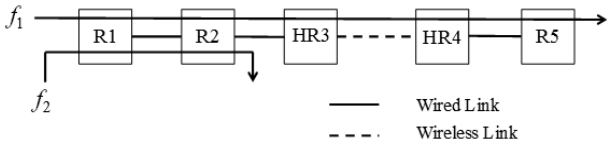


Fig. 5 A typical hybrid wired/wireless path.

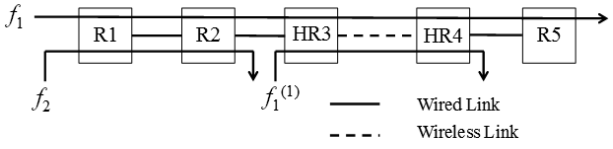


Fig. 6 Equivalent form of hybrid wired/wireless path.

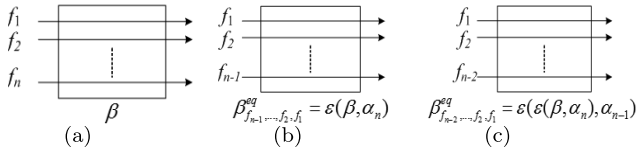


Fig. 7 Equivalent service curve computation for general case.

$$\begin{aligned}\beta^{eq}_{f_{n-1},...,f_2,f_1} &= \epsilon(\beta,\alpha_n)\\ &= \delta_{T+\frac{b_m}{R}+s} \otimes \gamma_{R\cdot s, R-r_n}, (s \geq 0)\end{aligned} \tag{3}$$

where s is an intermediate argument for computing the least upper delay bound, $\epsilon(\cdot,\cdot)$ is a function to compute the equivalent service curve [14]. And so on we can computer the equivalent service curve for the flows set $(f_1, f_2, ..., f_{n-2})$ as follows:

$$\beta^{eq}_{f_{n-2},...,f_2,f_1} = \epsilon(\epsilon(\beta,\alpha_n),\alpha_{n-1}) \tag{4}$$

Recursively we can get the equivalent service curve for flow f_1 as:

$$\beta^{eq}_{f_1} = \epsilon(...\epsilon(\epsilon(\beta,\alpha_n),\alpha_{n-1})...,\alpha_2) \tag{5}$$

Then the output arrival curve of f_1 can be derived as:

$$\alpha_1^* = \alpha_1 \oslash \beta^{eq}_{f_1} \tag{6}$$

and its delay bound is

$$D_{f_1} = H(\alpha_1, \beta^{eq}_{f_1}) \tag{7}$$

where $H(\cdot,\cdot)$ is the function to compute the maximum horizontal distance between the arrival curve and the service curve. By this method, we can compute the equivalent service curve, the output arrival curve and delay bound for any flow.

Based on the above derivation, we can analyze all the routers in Fig. 6 in turn. Firstly we compute the equivalent service curves for f_1 and f_2 at R1; then the output arrival curves of f_1 and f_2 at R1 can be computed, which are just the arrival curves at R2. The delay bounds of f_1 and f_2 at R1 can also be computed. Repeating the analyzing procedure until R5, the performance of the hybrid wired/wireless path

is achieved.

This analysis method can be easily extended to the case of on-chip network. We scan the routers in on-chip network iteratively. For a certain router, if all arrival curves are known, we compute the equivalent service curve, the output arrival curve and the delay bound for each flow. Otherwise this router is skipped in this round iteration. After all routers are scanned, one-round iteration is finished. Then we start another round iteration to scan the routers that are not computed in last iterations. This procedure will be repeated until all the routers have been computed. The algorithm is shown as follows:

Algorithm 1: Analyze each router in on-chip network
Input: V, F, $\{F_v|\forall v\}$
Output: $\{\beta_f^{v,eq}, \alpha_f^v, D_f^v|\forall v, \forall f\}$
Initial:
  $U = V$;
  foreach $v \in V$ **do**
    foreach $f \in F$ **do**
      if v is the source of f **then**
        $\alpha_f^v = \alpha_f$;
      else
        $\alpha_f^v = \varnothing$;
      end
    end
  end
While $U \neq \varnothing$ **do**
  foreach $v \in U$ **do**
    if $\forall f \in F_v, \alpha_f^v \neq \varnothing$ **then**
      foreach $f \in F_v$ **do**
        Compute $\beta_f^{v,eq}$ by Eq. (5);
        Compute α_f^{v+} by Eq. (6);
        Compute D_f^v by Eq. (7);
      end
      $U = U - v$;
    end
  end
end
return $\{\beta_f^{v,eq}, \alpha_f^v, D_f^v|\forall v, \forall f\}$

where $\beta_f^{v,eq}$ is the equivalent service curve of router v for flow f; α_f^v is the arrival curve of f at v; D_f^v is the delay bound of f at v and α_f^{v+} is the output arrival curve of f at v.

4.2 Wireless Link Allocation Algorithm

The goal of wireless link allocation is to replace the congested wired paths by wireless links so that the interconnect delay is reduced.

To measure the congestion of a certain path (e.g. from v_p to v_q), we define a congestion factor $b_f^{(p,q)}$:

$$b_f^{(p,q)} = \frac{\sum_{i=p}^{q} D_f^{v_i}}{HC(p,q)} \tag{8}$$

where f is a flow traversing this path; $HC(\cdot,\cdot)$ is the function to calculate the hop count from p to q. The physical meaning of $b_f^{(p,q)}$ is the delay increasing rate of f from v_p to v_q.

In the on-chip network, if $b_f^{(p,q)}$ is the largest congestion factor, it indicates that the path (v_p, v_q) is the most congested path. Therefore we should bypass the flow f by wireless link to avoid it entering (v_p, v_q) so that the congestion can be alleviated. Thus v_p should be the start point of the wireless link. For the end point, we choose the lightest-loaded downstream router which is traversed by f. Injecting flow f to the lightest-loaded router is to avoid an unintentional congestion caused by wireless link insertion. We use D_f^v to measure the router's load. Thus we choose the router with smallest D_f^v as the end point of wireless link. After the new wireless link is allocated, the topology and routing of on-chip network is changed. Then we should calculate new congestion factors to detect the new performance bottleneck. This procedure is repeated until the number of wireless links used across the chip reaches the wireless links budget.

Based on the above discussion, we propose an efficient heuristic algorithm for wireless link allocation. The algorithm is shown as follows:

Algorithm 2: Allocate wireless links
Input: $\mathcal{N}$, V, F, N_{wl}, $\{F_v|\forall v\}$, $\{D_f^v|\forall v, \forall f\}$
Output: $\mathcal{HN}$
While $N_{wl} \neq 0$ **do**
  foreach $f \in F$ **do**
    foreach $\left((x,y)|f \in F_{v_x} \cap F_{v_y}\right)$ **do**
      $b_f^{(x,y)} = \frac{\sum_{i=x}^{y} D_f^{v_i}}{HC(x,y)}$;
    end
  end
  $b_{f^*}^{(p,q)} = \max\left(b_f^{(x,y)}|\forall(x,y), \forall f\right)$;
  $D_{f^*}^{v_{q'}} = \min\left(D_{f^*}^{v_x}|\forall x, q \to x\right)$;
  Place Wireless Link between (p, q');
  Generate new $\mathcal{HN}$ and re-route the flows;
  Compute the retransmission-flows of Wireless Link;
  Compute $\{D_f^v|\forall v, \forall f\}$ for $\mathcal{HN}$ by Algorithm 1;
  $N_{wl} = N_{wl} - 1$;
end
return $\mathcal{HN}$

where $\max(\cdot)$ and $\min(\cdot)$ are the functions to get the maximum and minimum value respectively, $q \to x$ means v_x is v_q's downstream router.

The algorithm starts from the underlying wired on-chip network $\mathcal{N}$. For each communication flow f, the congestion factor $b_f^{(x,y)}$ of any path (v_x, v_y) which is traversed by f is computed. Then we can find the largest congestion factor $b_{f^*}^{(p,q)}$ over the network. Next we find the smallest $D_{f^*}^{q'}$ where $v_{q'}$ is v_q's downstream router. Then we place a wireless link between p and q'. Since a new wireless link is placed, a new hybrid on-chip network $\mathcal{HN}$ is generated. In the next step, the flows are re-routed and the retransmission flows are computed. Thereafter the delay bounds $\{D_f^v\}$ are computed. The previous procedure is repeated until all the

wireless links are placed.

In Algorithm 1 and Algorithm 2, there is no any topological assumption of the on-chip network. The performance analytical model and congestion factor are derived from a general case of hybrid wired/wireless path. The procedure of allocation algorithm is also topology independent. Therefore the proposed algorithms are applicable to any topology of on-chip network.

To analyze the complexity of the proposed wireless link allocation algorithm, we assume there are n routers and p flows in the on-chip network. For algorithm 1, the worst case is all p flows traverse all n routers. Then algorithm 1 has a complexity of $O(n \times p)$. For algorithm 2, the condition "foreach $\left((x, y)|f \in F_{v_x} \cap F_{v_y}\right)$ do" will run C_n^2 $(=\frac{n(n-1)}{2})$ times in worst case that the topology is a chain. Therefore algorithm 2 has a complexity of $O(N_{wl} \times p \times \frac{n(n-1)}{2})$. Hence the complexity of the wireless link allocation algorithm is $O(N_{wl} \times p \times n^2)$ in worst case.

5. Wireless Link Allocation Algorithm Extension to Worst Case Delay Guarantee

In some applications, especially real-time applications, there are explicit delay constraints for communications of IP cores. This means the on-chip network should provide the worst case delay guarantee; otherwise the applications may break down. For these cases, the problem of wireless link allocation is formulated as a constrained optimization:

Given:

- Total wireless links budget N_{wl};
- Application communication flows F and *arrive curve* α_f of each flow;
- *Service curve* β of each router;
- Routing algorithm $\mathcal{R}$;
- The underlying wired on-chip network $\mathcal{N}$;
- The delay-constrained flows F_C and the delay constraints of each flow $\hat{D}_f$;

Determine:

The particular pairs of routers (v_x, v_y) to place N_{wl} wireless links that can minimize the average packet delay D

Subject to:

$$D_f \leq \hat{D}_f, \ \forall f \text{ which is delay-constrained} \tag{9}$$

and *at most one* wireless link is added per router.

To resolve this problem, the key point is to estimate the worst case delay accurately. Based on the theory of *Network Calculus*, the least upper bound of delay in on-chip network can be calculated [14]. As mentioned above, we have already calculated the least upper delay bound of flow f at router v (D_f^v) by Algorithm 1. Therefore the end-to-end worst case delay of flow f can be calculated as

$$D_f = \sum_{v \in V_f} D_f^v \tag{10}$$

Based on this description, we can extend the wireless link allocation algorithm to worst case delay guarantee. The extended algorithm evaluate the worst case delay of the constrained flows first. If there are some delay-violated flows, it attempts to insert some wireless links to eliminate delay violation as much as possible. The basic rule of determining where to insert a wireless link is to compare the number of delay-violated flows that can be eliminate when this wireless link is inserted in different position and find out the most beneficial position. The overall extended algorithm (Algorithm 3) flow is shown in Fig. 8.

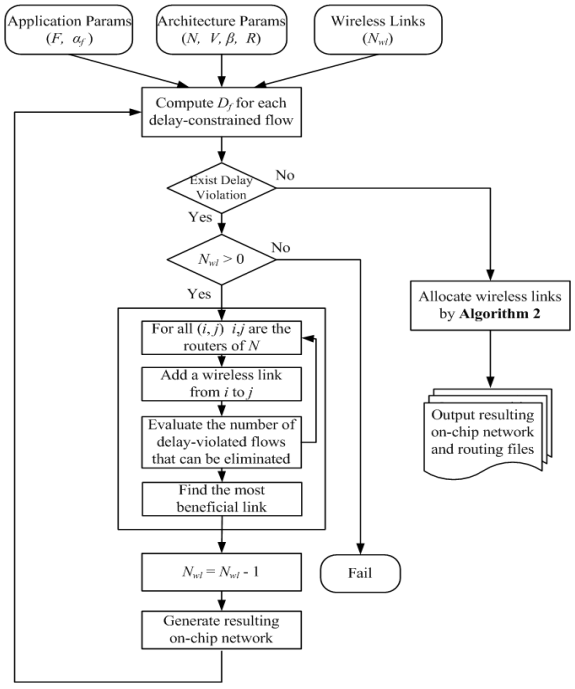


Fig. 8 Wireless link allocation algorithm extension to worst case delay guarantee.

The algorithm starts with a standard wired on-chip network $(\mathcal{N})$ and takes the application parameters (communication flows F and their arrival curve α_f), architecture parameters (routers V, service curve β and routing algorithm $\mathcal{R}$) and the amount of wireless links allowed to use (N_{wl}) as inputs. First, the algorithm computes the worst case delay (D_f) of each delay-constrained flow. If there exists delay violation (i.e. $\exists D_f > \hat{D}_f$), the algorithm inserts wireless links iteratively to eliminate the delay violation. More specifically, the algorithm selects all possible pairs of routers (i.e., $C(\|V\|, 2)$ pairs, where $\|V\|$ is the number of router in the on-chip network) and inserts wireless links between them. After inserting each wireless link, the resulting network is evaluated to find out the number of delay-violated flows that can be eliminated. After the most beneficial wireless link (which means this wireless link can eliminate the maximum number of delay-violated flows) is found, the information about this wireless link is stored and the amount of available wireless links updated. This procedure repeats until all delay-violated flows are eliminated or all available resources are used. If all wireless links are used and there is still some

Table 3 Simulation parameters.

Param.	Description
Clock frequency	2.5 GHz
Number of cores on chip	16
Baseline topology	4×4 2-D Mesh
Routing algorithm	Deterministic routing
Switching technique	Wormhole
Number of wireless links	10
Wired link bandwidth	2.5 Gbps
Wireless link bandwidth	10 Gbps [11]
BER of wireless link	10^{-9} [10]

delay violation, it means it is impossible to satisfy the delay constraints with the given wireless resources. We have to adjust the design goal. If there are some wireless links left after eliminating all delay violation, we use Algorithm 2 to allocate the remaining wireless links. Finally, the resulting hybrid on-chip network and routing files are output.

6. Simulation Results

In this section, we analyze the characteristics of the proposed wireless links allocation algorithm and evaluate the performance of the hybrid on-chip network designed by the proposed algorithm. The hybrid on-chip network is simulated by a cycle accurate simulator [15] which models the progress of data flits accurately per clock cycle. We use two traffic patterns: synthetic random traffic and E3S realistic traffic. The synthetic random traffic is generated randomly by the simulator and the E3S realistic traffic is extracted from audio-video benchmark of E3S benchmark suites.

We consider two performance metrics: delay and energy dissipation. Delay refers to the number of clock cycles between the injection of a packet header flit at the source node and the reception of the tail flit at the destination. Energy dissipation per bit is the average energy dissipated for transferring one bit of data from the source to destination node. For a simulation scenario, tens of thousands times simulation are carried out and we average the results of performance metrics.

We use a 4×4 2-D Mesh as our baseline topology, and on top of that, we construct the hybrid on-chip network with at most 10 wireless links. Table 3 summarizes a list of the simulation parameters we used.

6.1 Evaluation under Synthetic Random Traffic

In the first set of simulations, we consider the scenarios of synthetic random traffic. We randomly choose 16 pairs of source and destination cores. From each source core, a traffic flow with arrival curve $\alpha_f(t) = \gamma_{2,r}(t)$ is injected, which means that the core generates r packets every ten cycles on average and the burstness is two packets. For each router, the service curve is $\beta(t) = 1 \cdot (t-1)^+$, which represents that router forwards one packet per cycle per link and the processing delay is one cycle. For this traffic pattern, we use the proposed wireless link allocation algorithm to generate

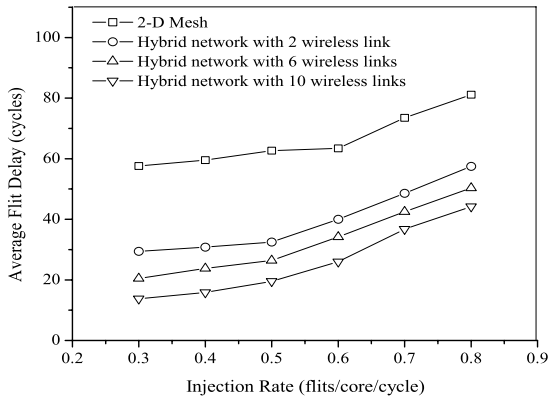


Fig. 9 Average flit delay with different flit injection rates.

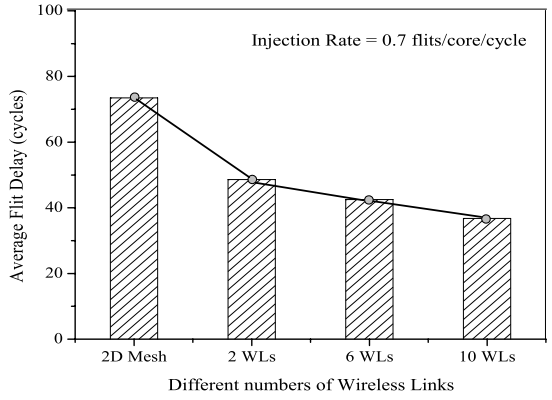


Fig. 10 Average flit delay with different numbers of wireless links.

the hybrid on-chip network with different numbers of wireless links. The 2-D mesh network is used as performance baseline.

Figure 9 shows the average flit delay plots as a function of injection rate (r). As can be seen, when the wireless links are inserted into the 2-D Mesh network, the average flit delay is significantly reduced. This performance gain is due to two reasons. Firstly the wireless link has comparative high bandwidth and it reduces the end-to-end hop counts. Therefore the average flit delay is shortened. Secondly our proposed algorithm is a congestion-aware method. It detects the most congested path in the on-chip network and places the wireless link to eliminate the congestion. This feature benefits not only the bypassed flows by the wireless links but also the flows transferred by the wired multi-hop paths.

It can also be observed that with increasing number of wireless links, the performance is improved but the improvement is more and more slight. In Fig. 9, the vertical distance between the curves is decreased from the top down. To show it more clearly, we compare the average flit delay with different numbers of wireless links at injection rate of 0.7 flits/core/cycle in Fig. 10. The average flit delay is reduced 25 cycles when inserting two wireless link into the wired mesh network. But it only reduced 6 cycles when increasing the number of wireless links from 6 to 10. This is because when the wireless links are sufficient enough to

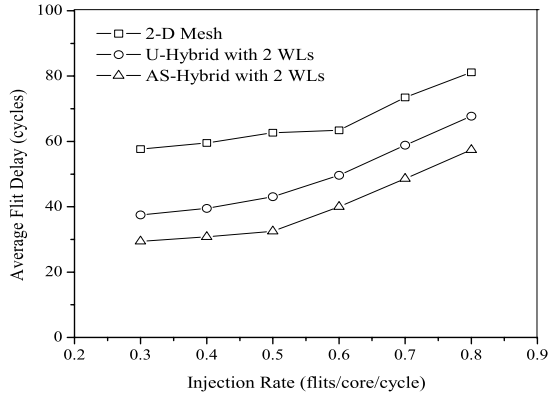


Fig. 11 Average flit delay comparison between application-specific and uniform hybrid on-chip network with 2 wireless links.

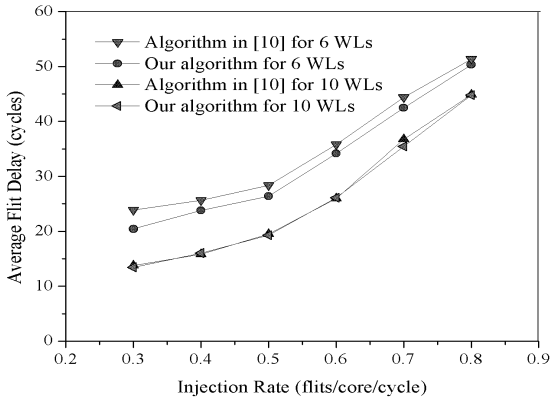


Fig. 12 Average flit delay comparison between our proposed algorithm and the algorithm in [11].

alleviate most network congestion, allocating more wireless links can only get little gain. Thus the designer can make a trade-off between the performance demands and area overhead of deploying wireless links.

We also compare the performance of application-specific hybrid on-chip network (denoted by AS-Hybrid) to a uniform hybrid on-chip network [11] (denoted by U-Hybrid). The U-hybrid is constructed based on the theory of "*small world*". The baseline topology is a ring which connects the routers and some wireless links are inserted between the selected routers by "*small world*" approach. Under the same traffic pattern and wireless links budget (2 wireless links), the performance comparison is shown in Fig. 11. As can be seen, the AS-Hybrid outperforms U-Hybrid. Although U-Hybrid employes the wireless links to replace multi-hop wired path, it is not aware of the traffic pattern and the congestion in the network. Therefore some heavily congested paths may still exist which exacerbate the network performance. In AS-Hybrid, the wireless links allocation algorithm guarantees to eliminate the congested paths in the order of congestion extent. Hence AS-Hybrid achieves better performance than U-Hybrid in a given application scenario.

In [11], the authors also proposed a simple traffic-dependent wireless link insertion algorithm as the extension to U-Hybrid. The probability of inserting a wireless link between routers i and j is

$$P_{ij} = \frac{h_{ij} \cdot f_{ij}}{\sum_{i,j} h_{ij} \cdot f_{ij}} \tag{11}$$

where f_{ij} is the frequency of communication between the ith source and jth destination; h_{ij} is the distance measured in the number of hops from the ith source and jth destination. To the best of our knowledge, this is the only algorithm for hybrid wired/wireless on-chip network which takes traffic pattern into account. Therefore we compare our proposed algorithm to this algorithm in the case of different wireless links budget. The average flit delay comparison is shown in Fig. 12. As can be seen, the AS-Hybrid outperforms traffic-dependent U-Hybrid in both cases. The reason is, in traffic-dependent U-hybrid, only traffic rate f_{ij} between source and destination is considered; it is not able to detect the congestion path which is caused by two contention flows. In contrast, our proposed algorithm does not care about the traffic rate from source to destination but the congested paths. It can also be observed that in the case of 10 wireless links, the two plots are really close. This result illustrates that when the wireless links are sufficient enough for the overall traffic, the allocation algorithm will become not important.

From the above results, it is clear that the application-specific hybrid on-chip network generated by the proposed algorithm outperform their corresponding uniform counterpart.

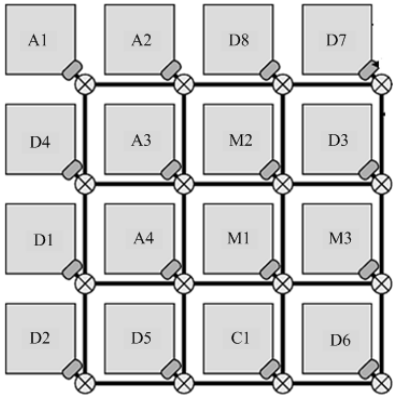


Fig. 13 16-core SoC for audio-video application. (A–ASIC, D–DSP, M–MEM, C–CPU)

6.2 Evaluation under Realistic Traffic

We also evaluate the performance of AS-Hybrid under real-world traffic pattern. The traffic pattern is extracted from audio-video benchmark of E3S benchmark suites [16]. We manually assign the tasks onto a 16-core SoC shown in Fig. 13. The traffic pattern is shown in Table 4. The wireless links budget is 4. We evaluate the performance of pure Mesh, U-Hybrid and AS-Hybrid in terms of both average flit delay and energy dissipation. In simulation, energy dis-

Table 4 Flow rates of core pairs in the audio-video application. (flits/core/cycle).

Pair	Rate	Pair	Rate	Pair	Rate
(A1,A2)	0.001	(A1,D8)	0.001	(A2,A1)	0.002
(A2,A3)	0.016	(A2,M2)	0.013	(A3,D4)	0.003
(A3,D8)	0.013	(A4,D1)	0.677	(A4,C1)	0.004
(D1,D2)	0.677	(D1,C1)	0.407	(D2,A2)	0.677
(D2,D1)	0.407	(D3,A4)	0.760	(D3,D5)	0.141
(D3,D6)	0.141	(D4,D1)	0.073	(D4,C1)	0.004
(D5,D6)	0.538	(D6,A2)	0.565	(D7,M2)	0.141
(D8,A1)	0.002	(D8,D7)	0.565	(M1,A4)	0.998
(M1,C1)	0.995	(M2,A3)	0.154	(M3,C1)	0.996
(C1,M1)	0.760	(C1,M3)	0.760		

Table 5 Performance under realistic traffic.

Architecture	Average Flit Delay (cycles)	Energy Consumption (pJ/bit)
2-D Mesh	31.62	1.226
U-Hybrid	30.43	1.150
AS-Hybrid	26.20	1.121

sipation per bit, E_{bit}, is calculated as follows:

$$E_{bit} = n^R_{hops} \times E^R_{bit} + n^L_{hops} \times E^L_{bit} + n^{WL}_{hops} \times E^{WL}_{bit} \qquad (12)$$

where n^R_{hops}, n^L_{hops} and n^{WL}_{hops} are the number of router, wired links and wireless links that the bit passes respectively; E^R_{bit} is the average energy consumption of transferring one bit of data through a router; E^L_{bit} and E^{WL}_{bit} are the average energy consumption of transferring one bit of data through a wired link and a wireless link of 1 mm length respectively. And in our parameters setting, $E^R_{bit} = 0.4$ pJ, $E^L_{bit} = 0.02$ pJ/mm and $E^{WL}_{bit} = 0.01$ pJ/mm [11].

The average flit delay and energy dissipation per bit are shown in Table 5. As can be seen, the performance of AS-Hybrid is better than U-Hybrid and pure Mesh in terms of both average flit delay and energy consumption per bit. And the performance improvement is even better than some cases of random traffic pattern. Indeed, the realistic traffic pattern is even more unbalanced compared to random traffic pattern in some cases, which results in heavier network congestion. Therefore the congestion-aware feature of AS-Hybrid is more beneficial than U-Hybrid and pure Mesh. This result shows that the proposed wireless links allocation algorithm is indeed beneficial for application-specific SoC design.

6.3 Evaluation for Worst Case Delay Guarantee

To evaluate the extended algorithm (Algorithm 3) for worst case delay guarantee, we also use the realistic traffic pattern in Table 4. We set the delay constraints as follows: the worst case delay from MEM cores to DSP or ASIC cores must be less than 30 cycles. These constraints are used to mimic the Memory Wall Effect. Since the Memory Wall always exacerbate the program performance greatly, there are usually the explicit time constraints to memory access. Therefore, four flows, (A2, M2), (M1, A4), (M2, A3) and (D7, M2), are delay-constrained. And the number of available wireless links is still 4.

Table 6 Worst case delay (cycles) comparison between 2-D mesh and AS-hybrid network.

Communicaton Flow	2-D Mesh	AS-Hybrid
(A2, M2)	16	12
(M1, A4)	27	23
(M2, A3)	18	15
(D7, M2)	48	9

Table 6 shows the worst case delay comparison between 2-D Mesh (before wireless links insertion) and AS-Hybrid network which is generated by Algorithm 3. We can see, before wireless links insertion, the worst case delay of flow (D7, M2) exceeds 30 cycles. The algorithm detected this delay violation and inserted the first wireless link between IP core D7 and M2. Then the worst case delay is decreased dramatically and satisfies the delay constraint. Then the algorithm allocated the other 3 wireless links and generated the final AS-Hybrid on-chip network. The results in Table 6 show that there is no any delay violation in the final AS-Hybrid on-chip network.

7. Conclusions

Hybrid wired/wireless on-chip network is a promising solution for the complex on-chip communication problems in multi-/many-core SoC design.

In this paper, we investigate how to design a effective hybrid on-chip network for application-specific SoC. We build a network performance analysis model based on *Network Calculus* and propose a heuristic congestion-aware wireless link allocation algorithm. This algorithm can achieve the optimum placement of the wireless links in the on-chip network according to the status of network congestion. The algorithm can also be extended to the worst delay guarantee problem. The simulation results show that the hybrid on-chip network which is designed by the proposed algorithm improves the performance in terms of both delay and energy consumption significantly.

In the future, we will revise the proposed algorithm to reduce the computational complexity.

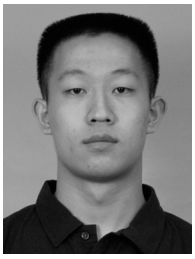
References

[1] L. Benini and G.D. Micheli, "Networks on chips: A new SoC paradigm," Computer, vol.35, no.1, pp.70–78, 2002.
[2] U.Y. Ogras and R. Marculescu, "It's a small world after All: NoC performance optimization via long-range link insertion," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol.14, no.7, pp.693–706, 2006.
[3] T. Krishna, A. Kumar, P. Chiang, M. Erez, and L.-S. Peh, "NoC with near-ideal express virtual channels using global-line communication," Proc. IEEE Symposium on High Performance Interconnects, pp.80–90, Aug. 2008.
[4] International Technology Roadmap for Semiconductors: http://www.itrs.net/links/2009ITRS/Home2009.htm
[5] K.K. O, K. Kim, B. Floyd, et al., "The feasibility of on-chip interconnection using antennas," Proc. IEEE/ACM International Conference on Computer-Aided Design (ICCAD'05), pp.421–426, 2005.

[6] P.J. Burke, S. Li, and Z. Yu, "Quantitative theory of nanowire and nanotube antenna performance," IEEE Trans. Nanotechnology, vol.5, no.4, pp.314–334, 2006.
[7] J. Lin, H.T. Wu, Y. Su, L. Gao, A. Sugavanam, J.E. Brewer, and K.K. O, "Communication using antennas fabricated in silicon integrated circuits," IEEE J. Solid-State Circuits, vol.42, no.8, pp.1678–1687, 2007.
[8] S. Deb, A. Ganguly, K. Chang, P. Pande, B. Beizer, and D. Heo, "Enhancing performance of network-on-chip architectures with millimeter-wave wireless interconnects," Proc. IEEE International Conference on Application-specific Systems, Architectures and Processors (ASAP 2010), pp.73–80, July 2010.
[9] D. Zhao and Y. Wang, "SD-MAC: Design and synthesis of a hardware-efficient collision-free QoS-aware MAC protocol for wireless network-on-chip," IEEE Trans. Comput., vol.57, no.9, pp.1230–1245, 2008.
[10] S. Lee, S. Tam, I. Perkianakis, S. Lu, M.F. Chang, C. Guo, G. Reinman, C. Peng, M. Naik, L. Zhang, and J. Cong, "A scalable micro wireless interconnect structure for CMPs," Proc. 15th International Conference on Mobile Computing and Networking (MobiCom'09), pp.217–228, Sept. 2009.
[11] A. Ganguly, K. Chang, S. Deb, P. Pande, B. Belzer, and C. Teuscher, "Scalable hybrid wireless network-on-chip architectures for multicore systems," IEEE Trans. Comput., vol.60, no.10, pp.1485–1502, 2010.
[12] J.Y. Le Boudec and P. Thiran, "Network calculus: A theory of deterministic queuing systems for the Internet," Lect. Notes Comput. Sci. 2050, Berlin, Germany, Springer-Verlag, 2004.
[13] H. Wang and J.B. Schmitt, "On the way to a wireless network calculus — The single node case with retransmission," Technical Report 375/10, University of Kaiserslautern, Germany, 2010.
[14] Y. Qian, Z. Lu, and W. Dou, "Analysis of worst-case delay bounds for on-chip packet-switching networks," IEEE Trans. Comput.-Aided Des. Integrated Circuits Syst., vol.29, no.5, pp.802–815, 2010.
[15] Y. Hu, S. Yin, L. Liu, and S. Wei, "A mixed-level modeling for network-on-chip infrastructure in SoC design," Proc. IEEE Asia Pacific Conference on Circuits and Systems, pp.911–914, 2010.
[16] E3S benmark, http://ziyang.eecs.umich.edu/~dickrp/e3s/

Shouyi Yin received the B.S., M.S. and Ph.D. degree in Electronic Engineering from Tsinghua University, China, in 2000, 2002 and 2005 respectively. He has worked in Imperial College London as a research associate. Currently, he is with Institute of Microelectronics at Tsinghua University as an associate professor. His research interests include mobile computing, wireless communications and SoC design.

Yang Hu received the B.S. degree in electronic engineering from Tianjin University. Tianjin, China, in 2007. He is currently working toward the M.S. degree in microelectronics at Tsinghua-Intel Joint Center of Advanced Mobile Computing Technology, Tsinghua University, Beijing, China. His current research interests include architecture and simulation of NoC and Wireless NoC.



Zhen Zhang received the B.S. degree in microelectronics from Tsinghua University, China, in 2010. He is currently working towards the Ph.D. degree in microelectronics at Tsinghua University. His current research interests include VLSI SoC design and reconfigurable computing.

Leibo Liu received the B.S. degree in electronic engineering from Tsinghua University, Beijing, China, in 1999 and the Ph.D. degree in Institute of Microelectronics, Tsinghua University, in 2004. He now serves as an associate professor in Institute of Microelectronics, Tsinghua University. His research interests include reconfigurable computing, mobile computing and VLSI DSP.

Shaojun Wei received Ph.D. degree from Faculte Polytechnique de Mons, Belgium, in 1991. He became a professor in Institute of Microelectronics of Tsinghua University in 1995. He is a senior member of Chinese Institute of Electronics (CIE). His main research interests include VLSI SoC design, EDA methodology, and communication ASIC design.