US 20180183721A1

(19) **United States**

(12) **Patent Application Publication** (10) **Pub. No.: US 2018/0183721 A1**

ROWLANDS et al. (43) **Pub. Date: Jun. 28, 2018**

(54) **INTERFACE VIRTUALIZATION AND FAST PATH FOR NETWORK ON CHIP**

(71) Applicant: **NetSpeed Systems, Inc.**, San Jose, CA (US)

(72) Inventors: **Joseph ROWLANDS**, San Jose, CA (US); **Joji PHILIP**, San Jose, CA (US); **Sailesh KUMAR**, San Jose, CA (US); **Nishant RAO**, San Jose, CA (US)

(21) Appl. No.: **15/903,425**

(22) Filed: **Feb. 23, 2018**

Related U.S. Application Data

(63) Continuation of application No. 15/829,749, filed on Dec. 1, 2017.

(60) Provisional application No. 62/429,695, filed on Dec. 2, 2016.

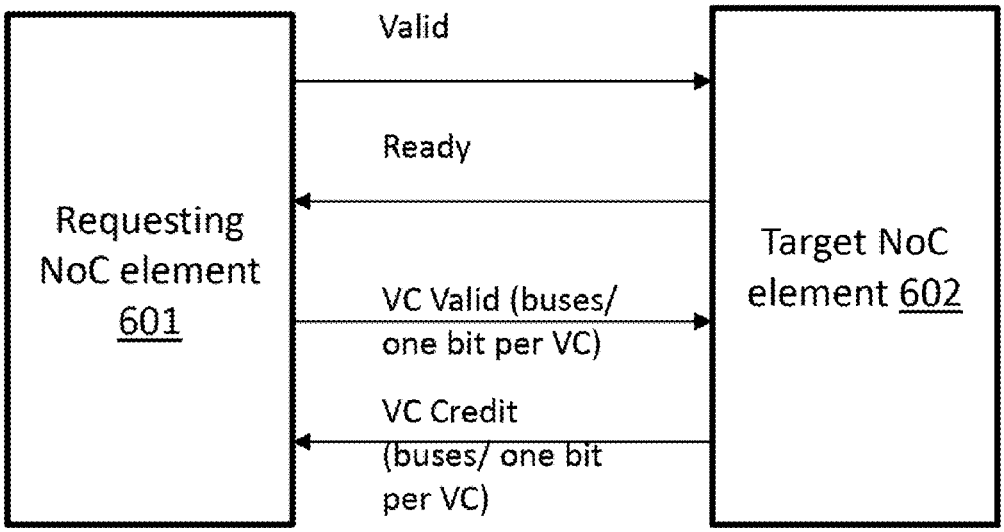
Publication Classification

(51) **Int. Cl.**
H04L 12/801 (2006.01)
H04L 12/933 (2006.01)

(52) **U.S. Cl.**
CPC ***H04L 47/39*** (2013.01); ***H04L 49/109*** (2013.01); *H04L 47/6275* (2013.01)

(57) **ABSTRACT**

Example implementations described herein are directed to a configurable Network on Chip (NoC) element that can be configured with a bypass that permits messages to pass through the NoC without entering the queue or arbitration. The configurable NoC element can also be configured to provide a protocol alongside the valid-ready protocol to facilitate valid-ready functionality across virtual channels.



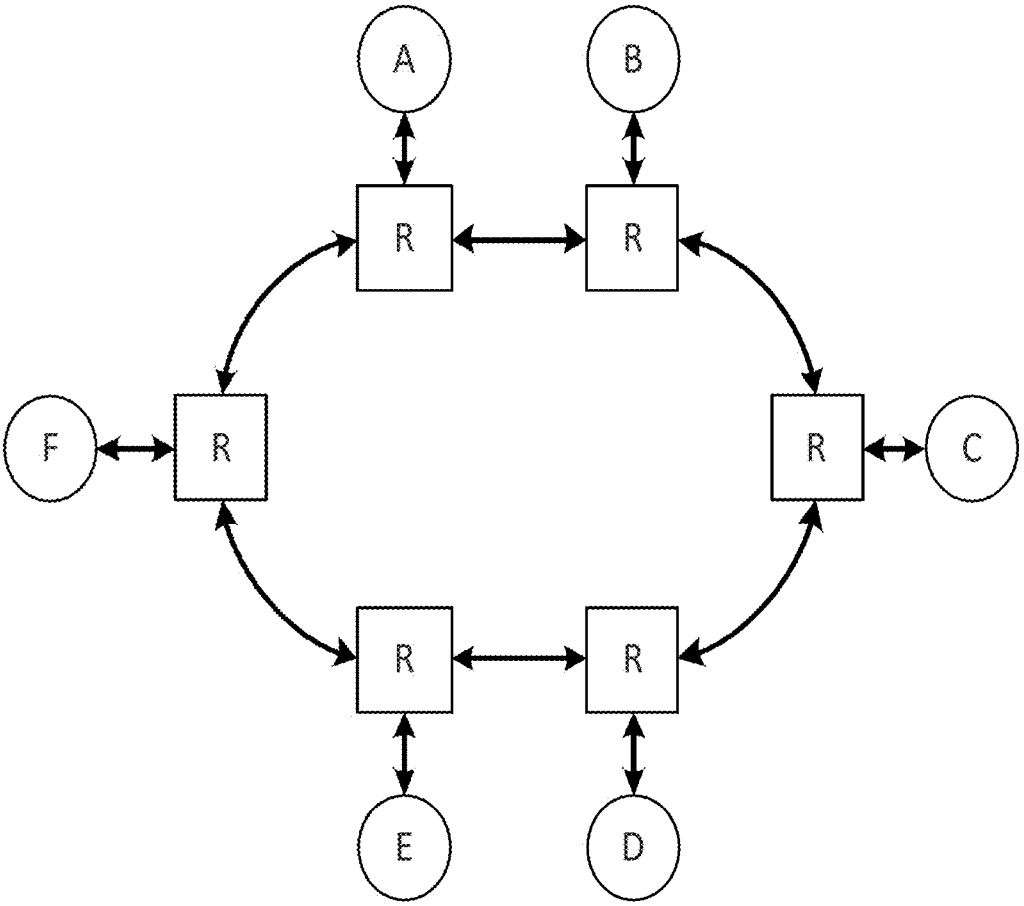


FIG. 1(a)

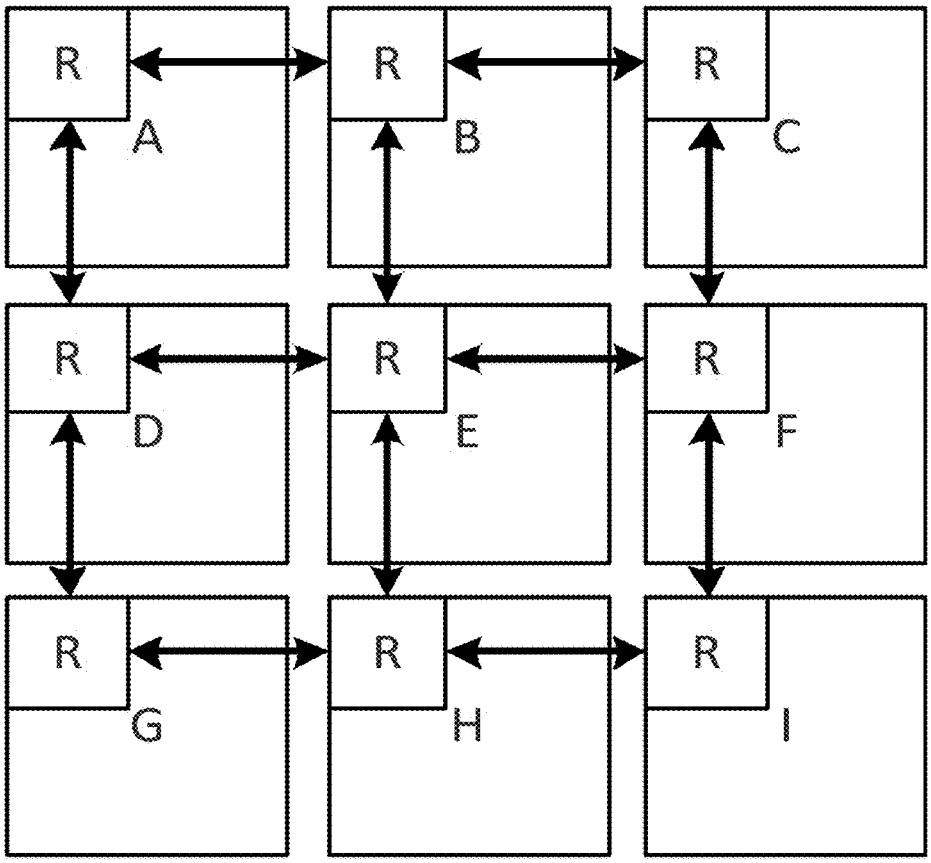


FIG. 1(b)

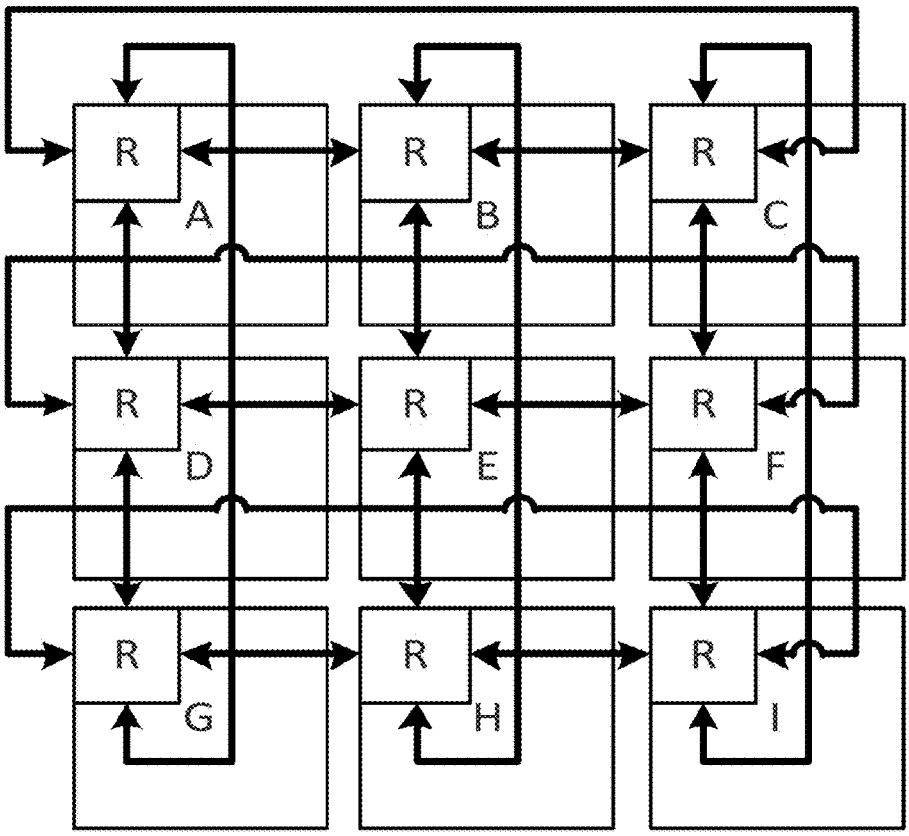


FIG. 1(c)

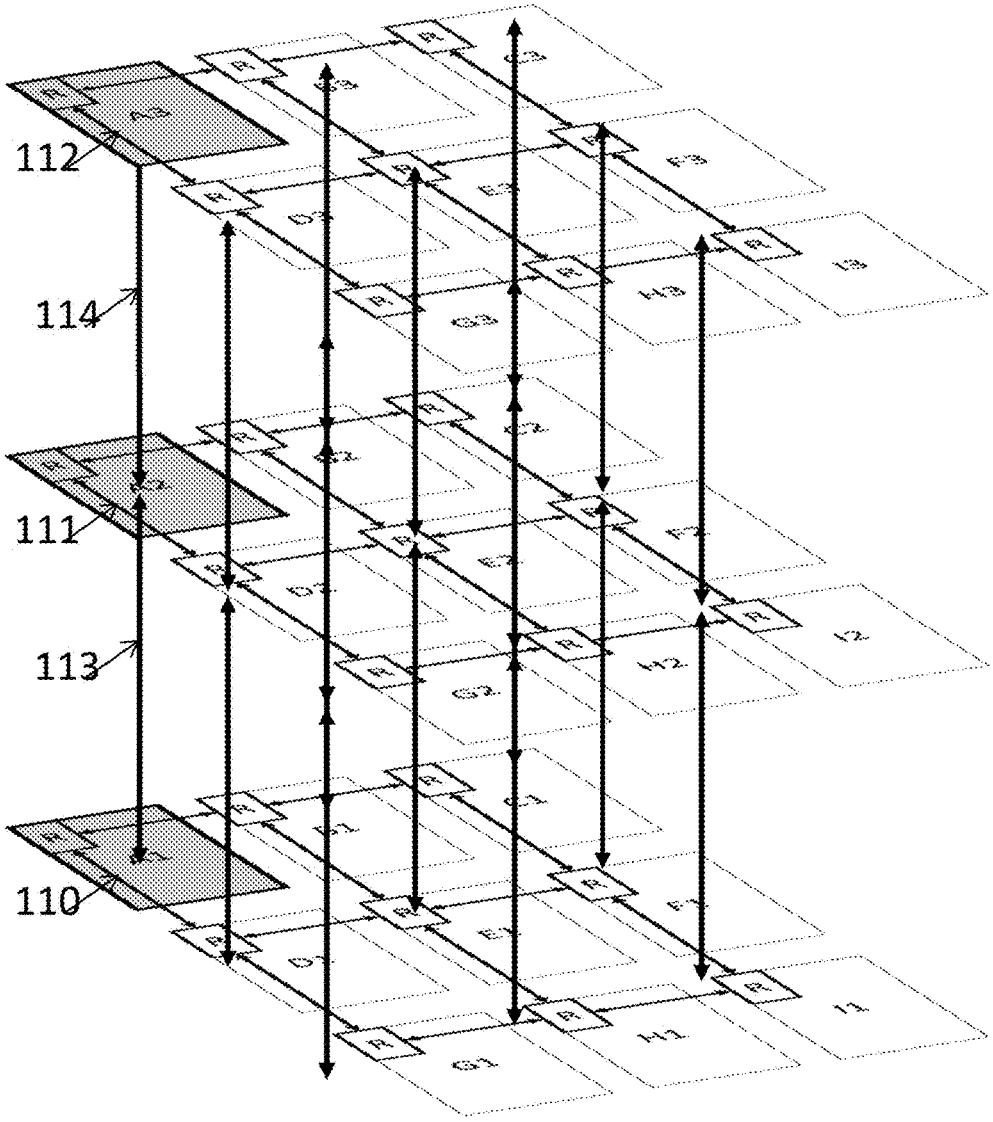


FIG. 1(d)

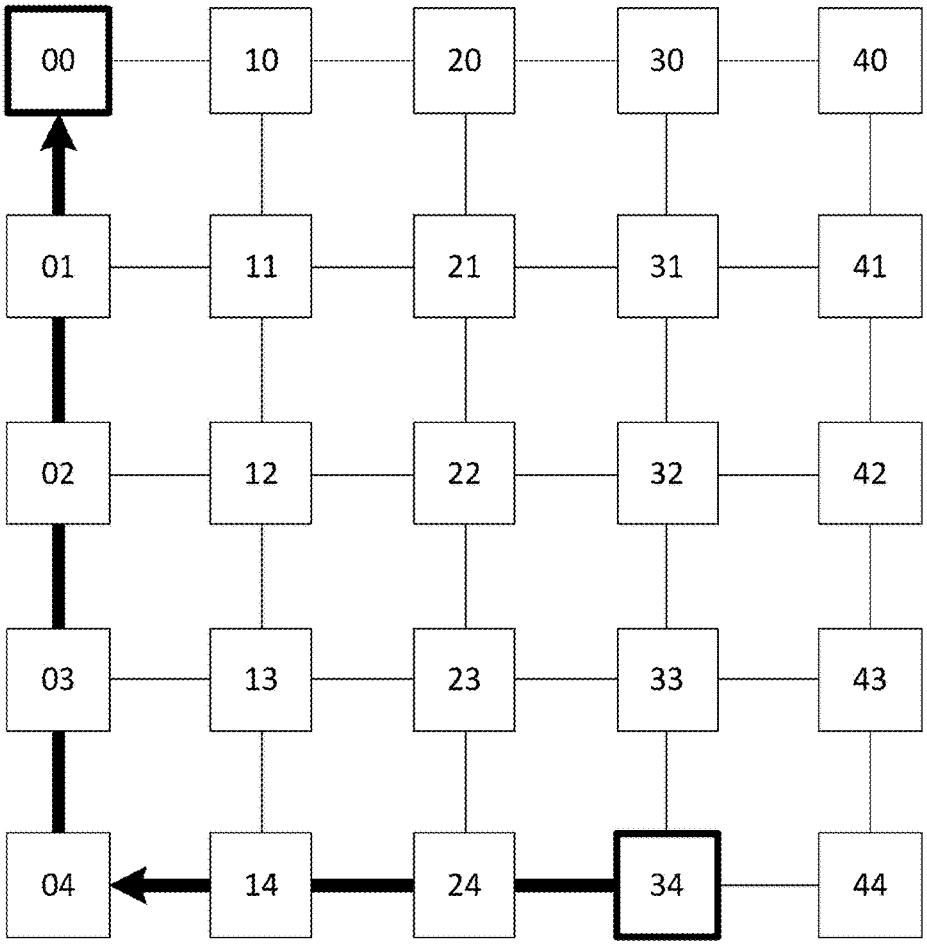

00
10
20
30
40
01
11
21
31
41
02
12
22
32
42
03
13
23
33
43
04
14
24
34
44

FIG. 2(A)

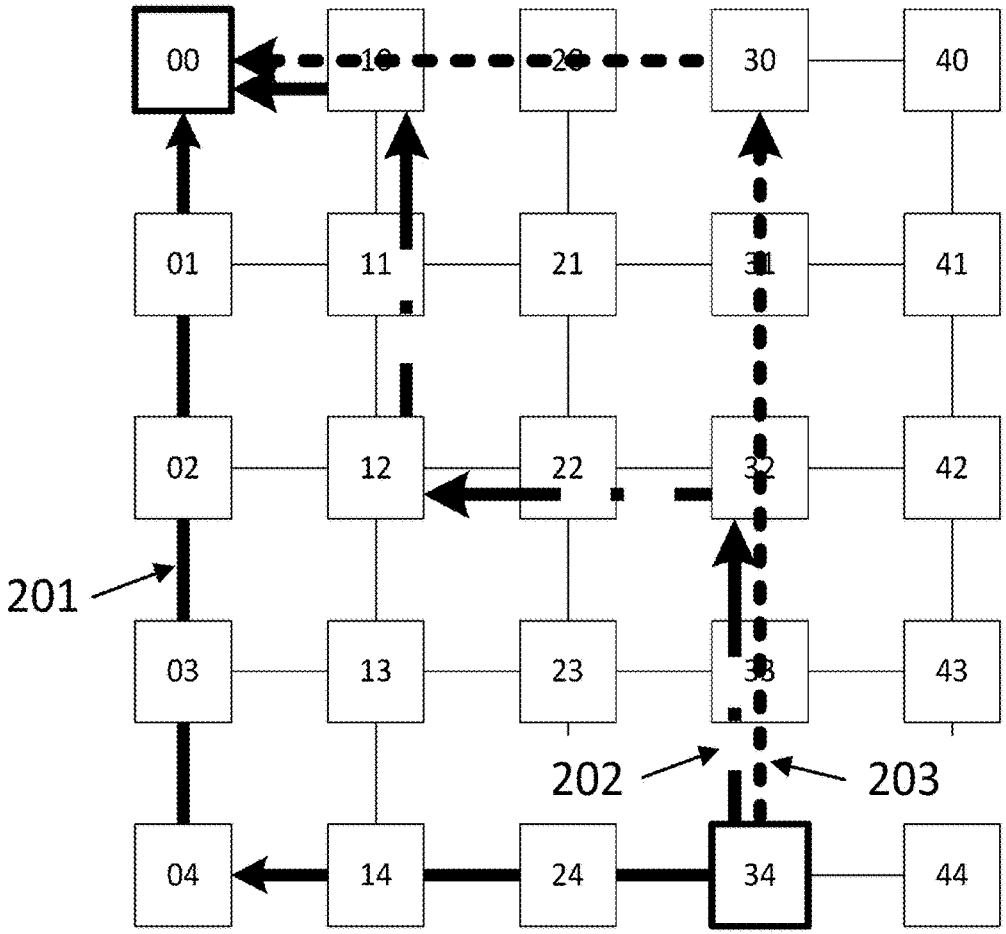

00
10
20
30
40
01
11
21
31
41
02
12
22
32
42
201
03
13
23
33
43
202
203
04
14
24
34
44

FIG. 2(B)

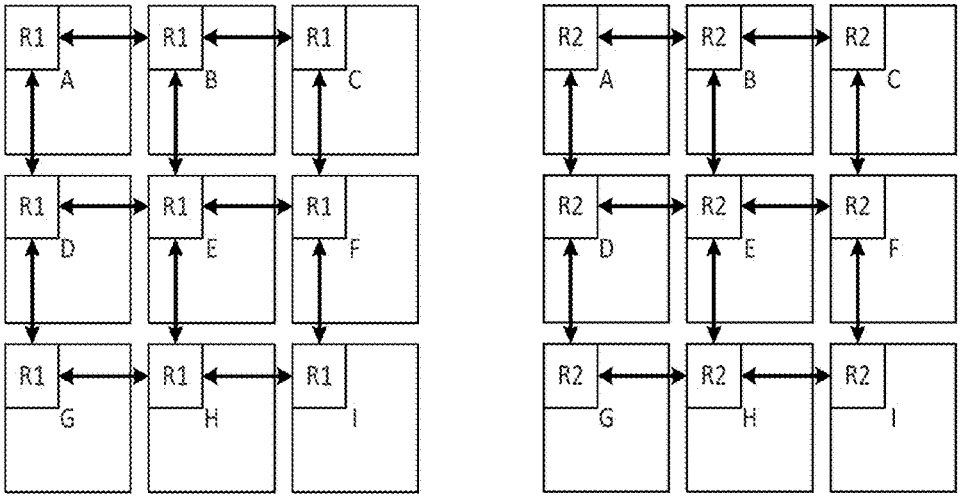

R1
A
R1
B
R1
C
R1
D
R1
E
R1
F
R1
G
R1
H
R1
I
R2
A
R2
B
R2
C
R2
D
R2
E
R2
F
R2
G
R2
H
R2
I

FIG. 3(a)

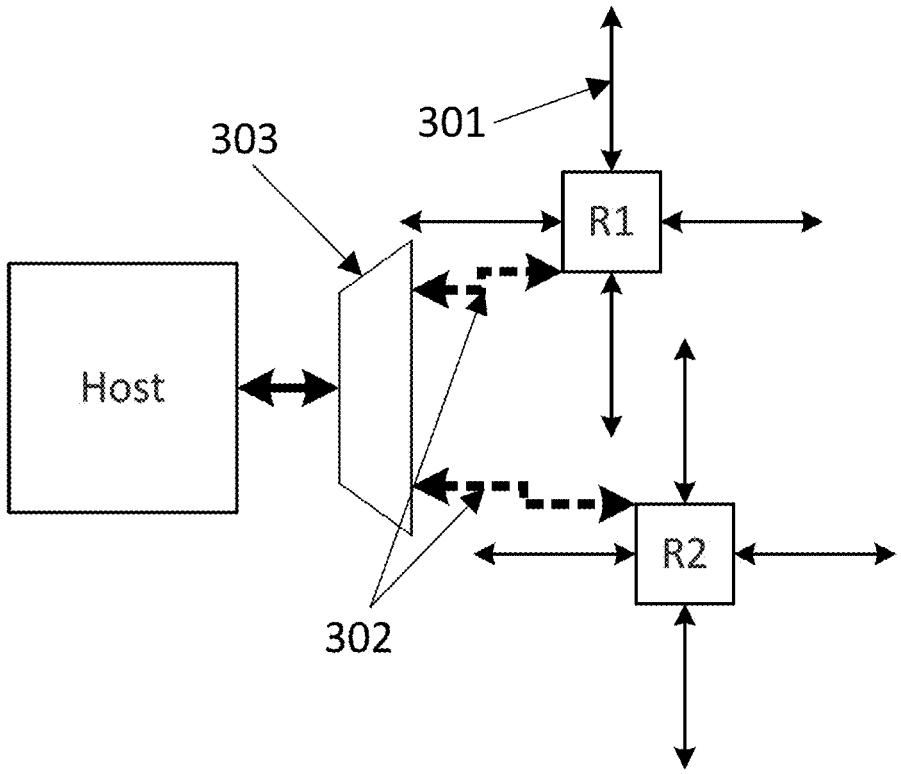


FIG. 3(b)

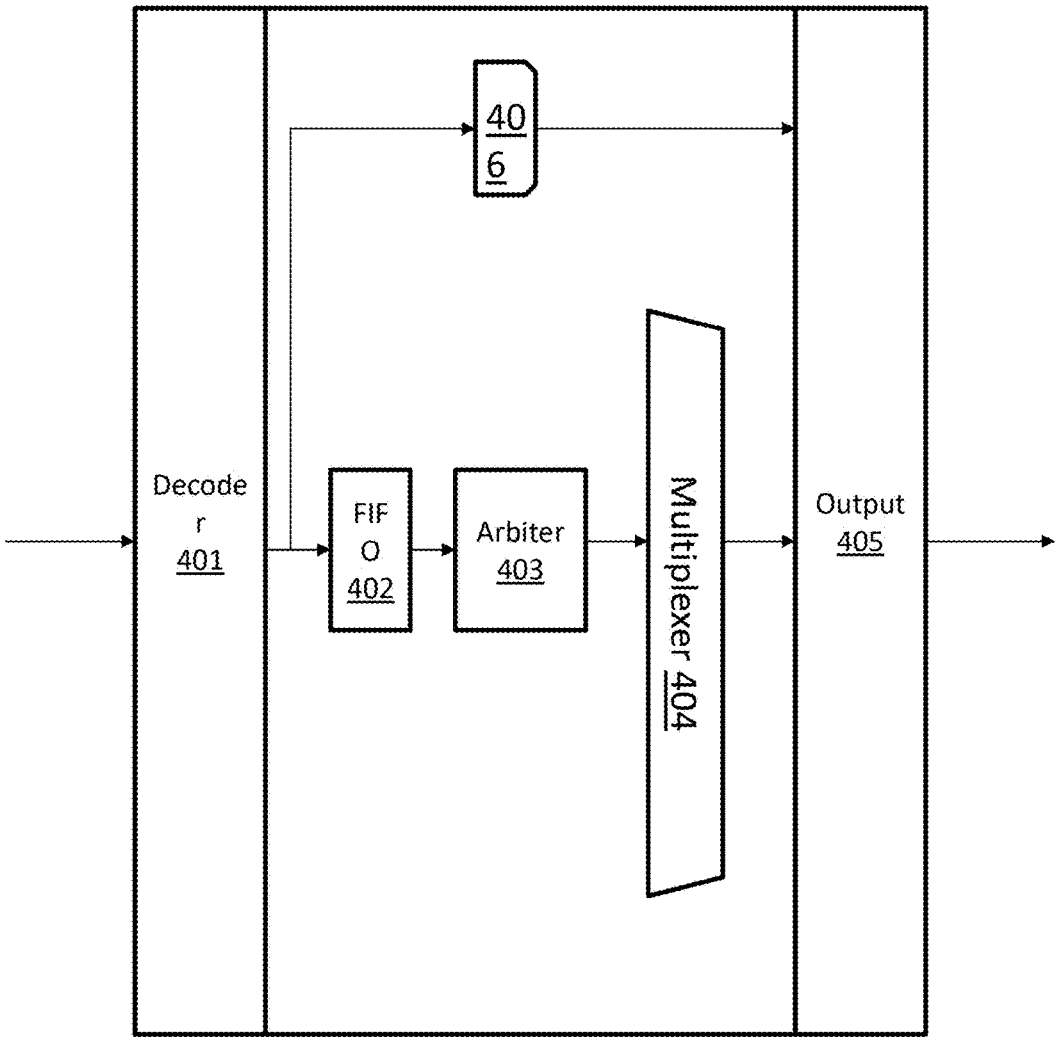

Decoder
401
FIFO
402
Arbiter
403
Multiplexer 404
406
Output
405

FIG. 4

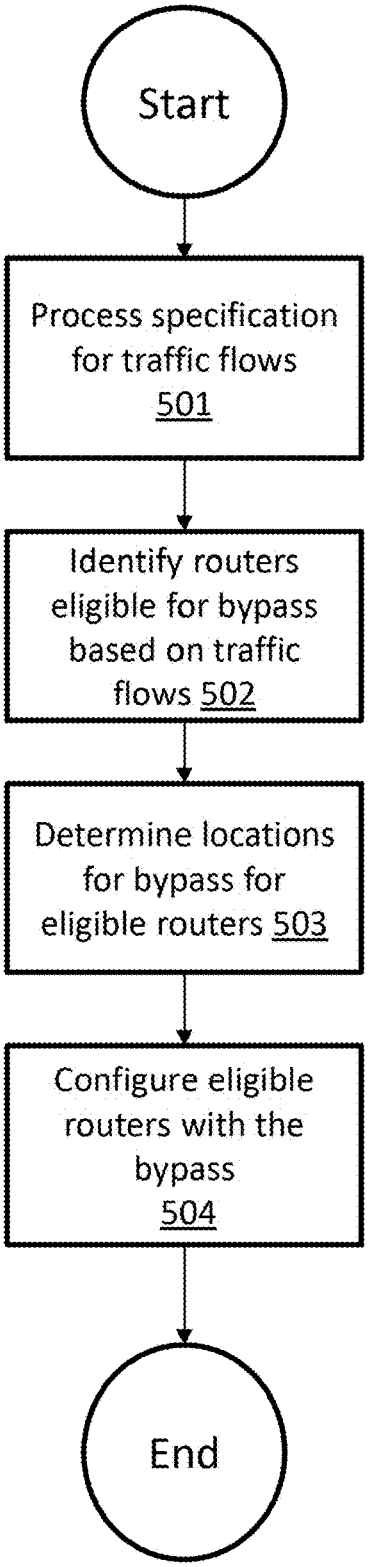

Start
Process specification for traffic flows 501
Identify routers eligible for bypass based on traffic flows 502
Determine locations for bypass for eligible routers 503
Configure eligible routers with the bypass 504
End

FIG. 5

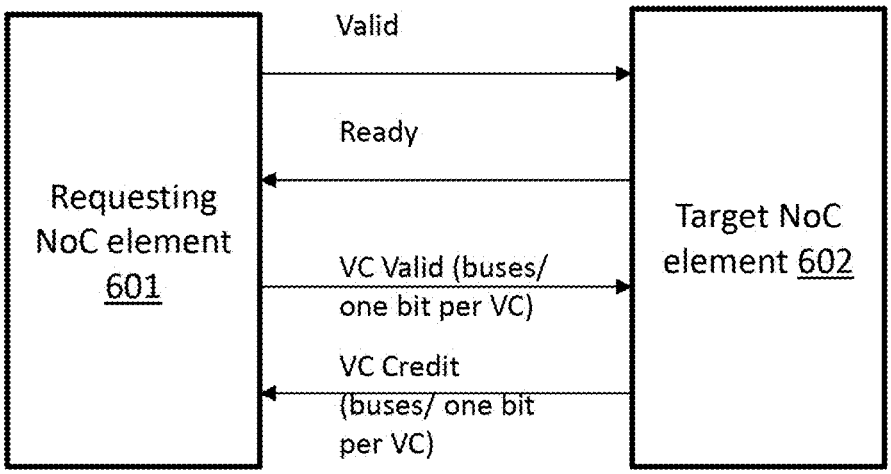

Valid
Ready
Requesting NoC element 601
Target NoC element 602
VC Valid (buses/ one bit per VC)
VC Credit (buses/ one bit per VC)

FIG. 6

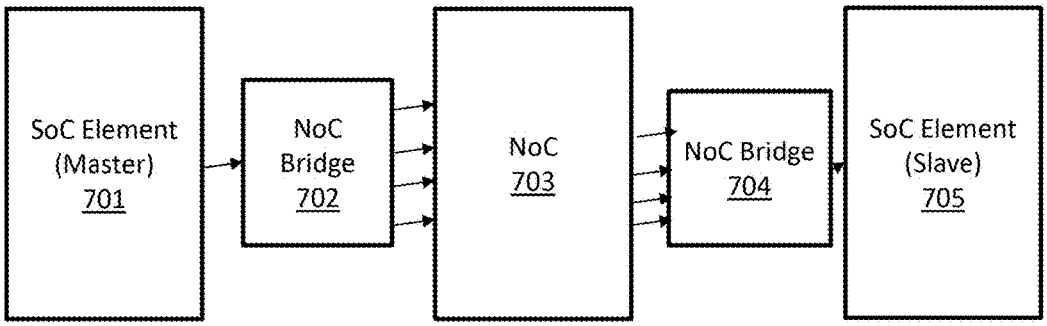

SoC Element
(Master)
701
NoC
Bridge
702
NoC
703
NoC Bridge
704
SoC Element
(Slave)
705

FIG. 7(a)

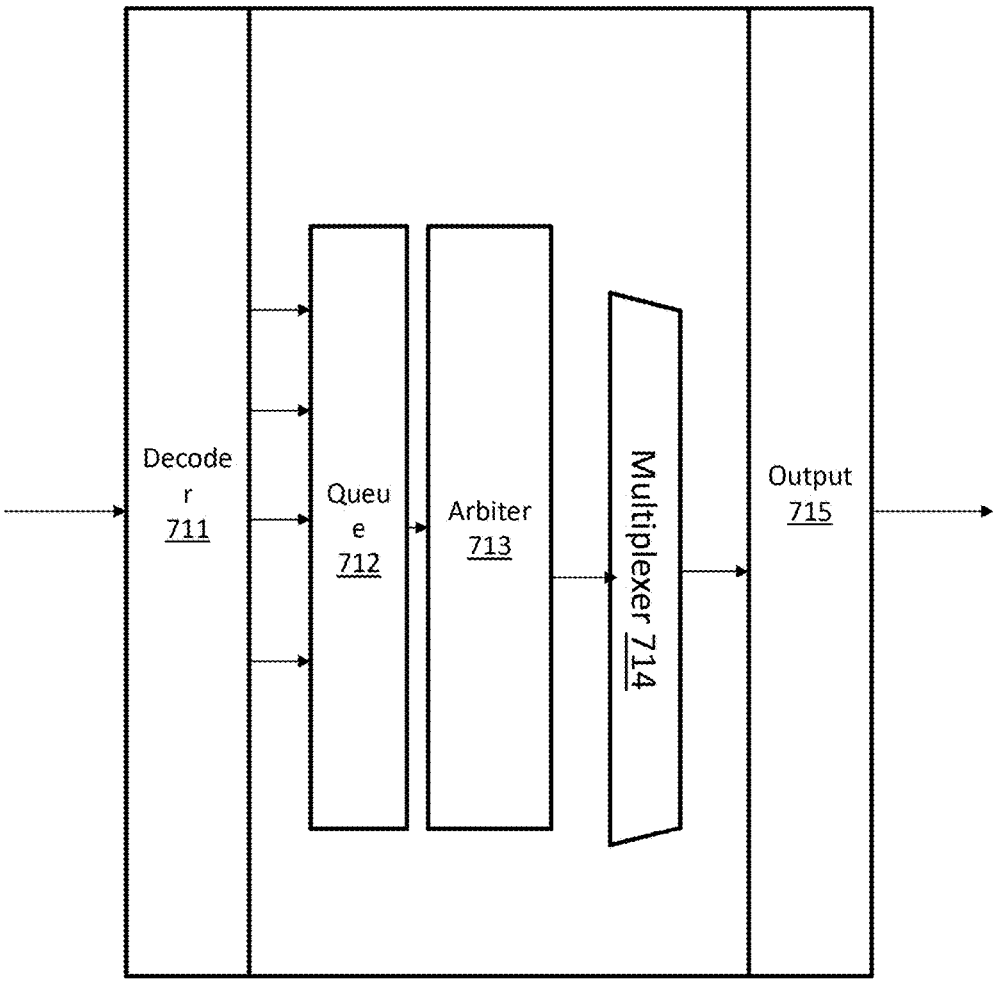

Decoder
711
Queue
712
Arbiter
713
Multiplexer 714
Output
715

FIG. 7(b)

Target VC	Ready	VC Credit
VC_Out_1	Yes	3
VC_Out_2	Yes	1
VC_Out_3	Yes	0
...	...	...

FIG. 8

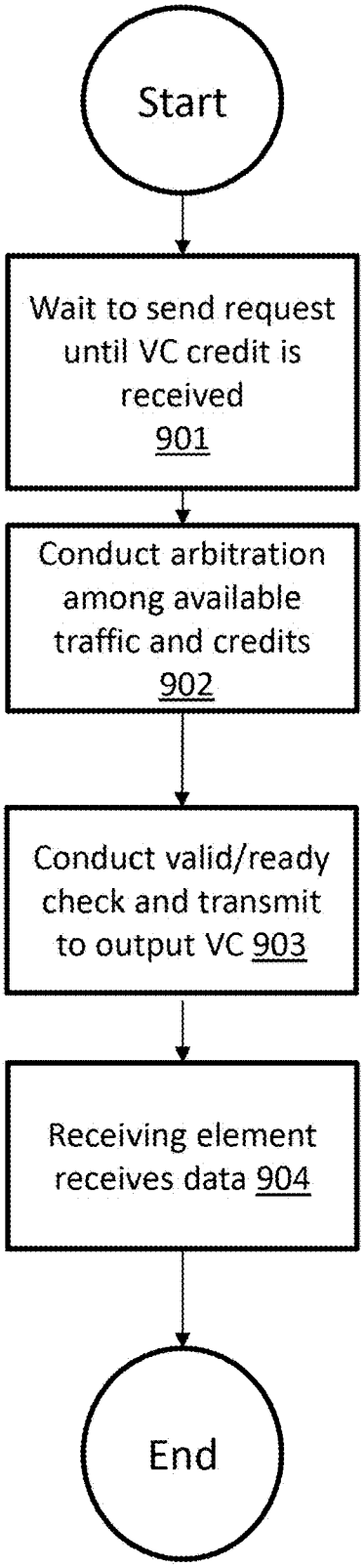

Start
Wait to send request until VC credit is received 901
Conduct arbitration among available traffic and credits 902
Conduct valid/ready check and transmit to output VC 903
Receiving element receives data 904
End

FIG. 9

INTERFACE VIRTUALIZATION AND FAST PATH FOR NETWORK ON CHIP

CROSS-REFERENCE TO RELATED APPLICATION

[0001] This regular U.S. patent application is a continuation of U.S. patent application Ser. No. 15/829,749, filed on Dec. 1, 2017 which is based on and claims the benefit of priority under 35 U.S.C. 119 from provisional U.S. patent application No. 62/429,695, filed on Dec. 2, 2016, the entire disclosure of which is incorporated by reference herein.

BACKGROUND

Technical Field

[0002] Methods and example implementations described herein are directed to interconnect architecture, and more specifically, to Network on Chip (NoC) architectures and the design and management thereof.

Related Art

[0003] The number of components on a chip is rapidly growing due to increasing levels of integration, system complexity and shrinking transistor geometry. Complex System-on-Chips (SoCs) may involve a variety of components e.g., processor cores, Digital Signal Processors (DSPs), hardware accelerators, memory and I/O, while Chip Multi-Processors (CMPs) may involve a large number of homogenous processor cores, memory and I/O subsystems. In both SoC and CMP systems, the on-chip interconnect plays a role in providing high-performance communication between the various components. Due to scalability limitations of traditional buses and crossbar based interconnects, Network-on-Chip (NoC) has emerged as a paradigm to interconnect a large number of components on the chip. NoC is a global shared communication infrastructure made up of several routing nodes interconnected with each other using point-to- point physical links.

[0004] Messages are injected by the source and are routed from the source node to the destination over multiple intermediate nodes and physical links. The destination node then ejects the message and provides the message to the destination. For the remainder of this application, the terms 'components', 'blocks', 'hosts' or 'cores' will be used interchangeably to refer to the various system components which are interconnected using a NoC. Terms 'routers' and 'nodes' will also be used interchangeably. Without loss of generalization, the system with multiple interconnected components will itself be referred to as a 'multi-core system'.

[0005] There are several topologies in which the routers can connect to one another to create the system network. Bi-directional rings (as shown in FIG. **1**(*a*)), 2D (two dimensional) mesh (as shown in FIGS. **1**(*b*)) and 2-D Taurus (as shown in FIG. **1**(*c*)) are examples of topologies in the related art. Mesh and Taurus can also be extended to 2.5-D (two and half dimensional) or 3-D (three dimensional) organizations. FIG. **1**(*d*) shows a 3D mesh NoC, where there are three layers of 3×3 2D mesh NoC shown over each other. The NoC routers have up to two additional ports, one connecting to a router in the higher layer, and another connecting to a router in the lower layer. Router **111** in the middle layer of the example has both ports used, one connecting to the router at the top layer and another connecting to the router at the bottom layer. Routers **110** and **112** are at the bottom and top mesh layers respectively, therefore they have only the upper facing port **113** and the lower facing port **114** respectively connected.

[0006] Packets are message transport units for intercommunication between various components. Routing involves identifying a path composed of a set of routers and physical links of the network over which packets are sent from a source to a destination. Components are connected to one or multiple ports of one or multiple routers; with each such port having a unique ID. Packets carry route information such as the destination's router and port ID for use by the intermediate routers to route the packet to the destination component.

[0007] Examples of routing techniques include deterministic routing, which involves choosing the same path from A to B for every packet. This form of routing is independent from the state of the network and does not load balance across path diversities, which might exist in the underlying network. However, such deterministic routing may implemented in hardware, maintains packet ordering and may be rendered free of network level deadlocks. Shortest path routing may minimize the latency as such routing reduces the number of hops from the source to the destination. For this reason, the shortest path may also be the lowest power path for communication between the two components. Dimension-order routing is a form of deterministic shortest path routing in 2D, 2.5-D, and 3-D mesh networks. In this routing scheme, messages are routed along each coordinates in a particular sequence until the message reaches the final destination. For example in a 3-D mesh network, one may first route along the X dimension until it reaches a router whose X-coordinate is equal to the X-coordinate of the destination router. Next, the message takes a turn and is routed in along Y dimension and finally takes another turn and moves along the Z dimension until the message reaches the final destination router. Dimension ordered routing may be minimal turn and shortest path routing.

[0008] FIG. **2**(*a*) pictorially illustrates an example of XY routing in a two dimensional mesh. More specifically, FIG. **2**(*a*) illustrates XY routing from node '**34**' to node '**00**'. In the example of FIG. **2**(*a*), each component is connected to only one port of one router. A packet is first routed over the x-axis till the packet reaches node '**04**' where the x-coordinate of the node is the same as the x-coordinate of the destination node. The packet is next routed over the y-axis until the packet reaches the destination node.

[0009] In heterogeneous mesh topology in which one or more routers or one or more links are absent, dimension order routing may not be feasible between certain source and destination nodes, and alternative paths may have to be taken. The alternative paths may not be shortest or minimum turn.

[0010] Source routing and routing using tables are other routing options used in NoC. Adaptive routing can dynamically change the path taken between two points on the network based on the state of the network. This form of routing may be complex to analyze and implement.

[0011] A NoC interconnect may contain multiple physical networks. Over each physical network, there may exist multiple virtual networks, wherein different message types are transmitted over different virtual networks. In this case, at each physical link or channel, there are multiple virtual channels; each virtual channel may have dedicated buffers at

both end points. In any given clock cycle, only one virtual channel can transmit data on the physical channel.

[0012] The physical channels are shared into a number of independent logical channels called virtual channels (VCs). VCs provide multiple independent paths to route packets, however they are time-multiplexed on the physical channels. A virtual channel holds the state needed to coordinate the handling of the flits of a packet over a channel. At a minimum, this state identifies the output channel of the current node for the next hop of the route and the state of the virtual channel (idle, waiting for resources, or active). The virtual channel may also include pointers to the flits of the packet that are buffered on the current node and the number of flit buffers available on the next node.

[0013] NoC interconnects may employ wormhole routing, wherein, a large message or packet is broken into small pieces known as flits (also referred to as flow control digits). The first flit is the header flit, which holds information about this packet's route and key message level info along with payload data and sets up the routing behavior for all subsequent flits associated with the message. Optionally, one or more body flits follows the head flit, containing the remaining payload of data. The final flit is the tail flit, which in addition to containing the last payload also performs some bookkeeping to close the connection for the message. In wormhole flow control, virtual channels are often implemented.

[0014] The term "wormhole" plays on the way messages are transmitted over the channels: the output port at the next router can be so short that received data can be translated in the head flit before the full message arrives, thereby facilitating the sending of the packet to the next router before the packet is fully received. This allows the router to quickly set up the route upon arrival of the head flit and then opt out from the rest of the conversation. Since a message is transmitted flit by flit, the message may occupy several flit buffers along its path at different routers so that the packet can exist in multiple routers, thereby creating a worm-like image.

[0015] Based upon the traffic between various end points, and the routes and physical networks that are used for various messages, different physical channels of the NoC interconnect may experience different levels of load and congestion. The capacity of various physical channels of a NoC interconnect is determined by the width of the channel (number of physical wires) and the clock frequency at which it is operating. Various channels of the NoC may operate at different clock frequencies, and various channels may have different widths based on the bandwidth requirement at the channel. The bandwidth requirement at a channel is determined by the flows that traverse over the channel and their bandwidth values. Flows traversing over various NoC channels are affected by the routes taken by various flows. In a mesh or Taurus NoC, there may exist multiple route paths of equal length or number of hops between any pair of source and destination nodes. For example, in FIG. **2**(*b*), in addition to the standard XY route between nodes **34** and **00**, there are additional routes available, such as YX route **203** or a multi-turn route **202** that makes more than one turn from source to destination.

[0016] In a NoC with statically allocated routes for various traffic flows, the load at various channels may be controlled by intelligently selecting the routes for various flows. When a large number of traffic flows and substantial path diversity is present, routes can be chosen such that the load on all NoC channels is balanced nearly uniformly, thus avoiding a single point of bottleneck. Once routed, the NoC channel widths can be determined based on the bandwidth demands of flows on the channels. Unfortunately, channel widths cannot be arbitrarily large due to physical hardware design restrictions, such as timing or wiring congestion. There may be a limit on the maximum channel width, thereby putting a limit on the maximum bandwidth of any single NoC channel.

[0017] Additionally, wider physical channels may not help in achieving higher bandwidth if messages are short. For example, if a packet is a single flit packet with a 64-bit width, then no matter how wide a channel is, the channel will only be able to carry 64 bits per cycle of data if all packets over the channel are similar. Thus, a channel width is also limited by the message size in the NoC. Due to these limitations on the maximum NoC channel width, a channel may not have enough bandwidth in spite of balancing the routes.

[0018] To address the above bandwidth concern, multiple parallel physical NoCs may be used. Each NoC may be called a layer, thus creating a multi-layer NoC architecture. Hosts inject a message on a NoC layer; the message is then routed to the destination on the NoC layer, where it is delivered from the NoC layer to the host. Thus, each layer operates more or less independently from each other, and interactions between layers may only occur during the injection and ejection times. FIG. **3**(*a*) illustrates a two layer NoC. Here the two NoC layers are shown adjacent to each other on the left and right, with the hosts connected to the NoC replicated in both left and right diagrams. A host is connected to two routers in this example—a router in the first layer shown as R**1**, and a router is the second layer shown as R**2**. In this example, the multi-layer NoC is different from the 3D NoC, i.e. multiple layers are on a single silicon die and are used to meet the high bandwidth demands of the communication between hosts on the same silicon die. Messages do not go from one layer to another. For purposes of clarity, the present disclosure will utilize such a horizontal left and right illustration for multi-layer NoC to differentiate from the 3D NoCs, which are illustrated by drawing the NoCs vertically over each other.

[0019] In FIG. **3**(*b*), a host connected to a router from each layer, R**1** and R**2** respectively, is illustrated. Each router is connected to other routers in its layer using directional ports **301**, and is connected to the host using injection and ejection ports **302**. A bridge-logic **303** may sit between the host and the two NoC layers to determine the NoC layer for an outgoing message and sends the message from host to the NoC layer, and also perform the arbitration and multiplexing between incoming messages from the two NoC layers and delivers them to the host.

[0020] In a multi-layer NoC, the number of layers needed may depend upon a number of factors such as the aggregate bandwidth requirement of all traffic flows in the system, the routes that are used by various flows, message size distribution, maximum channel width, etc. Once the number of NoC layers in NoC interconnect is determined in a design, different messages and traffic flows may be routed over different NoC layers. Additionally, one may design NoC interconnects such that different layers have different topologies in number of routers, channels and connectivity. The

channels in different layers may have different widths based on the flows that traverse over the channel and their bandwidth requirements.

[0021] In a NoC interconnect, if the traffic profile is not uniform and there is a certain amount of heterogeneity (e.g., certain hosts talking to each other more frequently than the others), the interconnect performance may depend on the NoC topology and where various hosts are placed in the topology with respect to each other and to what routers they are connected to. For example, if two hosts talk to each other frequently and require higher bandwidth than other interconnects, then they should be placed next to each other. This will reduce the latency for this communication which thereby reduces the global average latency, as well as reduce the number of router nodes and links over which the higher bandwidth of this communication must be provisioned.

[0022] A NoC uses a shared network to pass traffic between different components. Any particular traffic flow might cross multiple routers before arriving at its destination. While the NoC can be efficient in terms of sharing wires, there can be an adverse effect on latency. Each router needs to arbitrate between its various inputs ports to decide which packet will be sent in a cycle. After the arbitration, the data must be selected through a multiplexing (muxing) structure. This process can take one or more cycles to complete, depending on the microarchitecture of the routers and the frequency. This means that for each router a traffic flow must cross, it can be incurring additional cycles of delay. Wire delay between routers can also cause delay.

[0023] To reduce latency, the routers can be built with bypass paths that allow skipping some or all of the arbitration and muxing costs of a router. These bypass paths can be used opportunistically when the router is idle, or they can support a simpler arbitration that allows a significant decrease in cycle time loss. Intelligent use of bypasses in a system can improve average latency of requests.

[0024] Longer latency can hurt the performance of the system. Reducing the latency of traffic flows is an important goal. The benefit of lower latency vary between different traffic flows. Some components are very latency sensitive, where each additional cycle of latency can have a significant performance reduction. Other flows will be less sensitive to latency. Intelligent setup of the bypasses can select the traffic flows that will provide the largest overall benefit to the system performance.

[0025] When packets finish traversing a NoC, they arrive at the interface to a component. Because a NoC can have many different kinds of traffic, design of the interface can have a big impact on performance. Many interface protocols use a method of flow control that doesn't distinguish between the contents of the packets. This can create head-of-line blocking issues, where a more important packet is stuck behind a less important packet.

[0026] The destination component can often benefit from distinguishing between different incoming traffic flows, allowing it to accept the more important flows and hold off the less important flows when resources are scares. Support of an enhanced interface can allow the destination component to signal the network which traffic flows it is willing to accept. The network can then choose which packets to send, avoiding the head-of-line blocking issue.

[0027] The enhanced interface flow control can be coupled with the networks use of virtual or physical channels to further avoid head-of-line blocking. If lower priority packets are transported in a separate channel from the higher priority packets, the destination component can backpressure one channel and allow the other to continue unimpeded.

SUMMARY

[0028] Therefore, to address the aforementioned problems, there is a need for systems, methods, and non-transitory computer readable mediums to facilitate an opportunistic bypass system for a NoC, as well as a VC valid and credit system to facilitate the management of VCs of the NoC.

[0029] Aspects of the present disclosure involve a Network on Chip (NoC) having a plurality of channels and a valid-ready system with VC valid and VC credit going back, element configured to send a valid signal with a VC valid signal.

[0030] Aspects of the present disclosure further involve a network on chip (NoC) element involving a plurality of physical links and virtual links, and a configurable bypass between virtual links, and bypass logic configured to bypass the queue and the logic of the NoC element.

[0031] The bypass is configured to bypass the queue and the logic of the NoC element in an opportunistic manner in accordance with the desired implementation. The NoC can also involve a configurable router that has complete configurability in terms of which bypasses are available. The configurable router has output ports, in which any select input port can connect to an output port with a direct bypass.

[0032] Aspects of the present disclosure can further include methods and computer readable mediums directed to determining the selection of bypasses for NoC construction. Such methods and computer readable mediums can include algorithms that during NoC construction, create additional opportunities for bypassing. Such algorithms can include restrictions to bypass placement (e.g., connections requiring upsizing and downsizing do not have bypass) reshaping the NoC topology to create more links for the bypass, building the NoC to have equal number of ports with no clock crossing, and avoiding upsizing and downsizing links.

[0033] In example implementations, the algorithms for the creation of bypass paths can involve determining the possible bypass opportunities for the configurations based on restrictions, for each bypass opportunity, choosing which inputs go to the output based on calculation of expected traffic flows/bandwidth that are expected to have biggest impact on the specification (e.g., weighted average of traffic, also take latency and importance of traffic into consideration), and selecting the bypasses with the biggest benefit.

[0034] In example implementations, there can be algorithms such as a multiplexer selection algorithm to select which multiplexer to use (e.g., preselected versus post selected), opportunistic bypass processing (e.g., messages are sent through bypass if bypass is idle or if bypass is possible, bypass conducted based on latency and First In First Out (FIFO) depth).

[0035] In example implementations, there can be NoC elements and configuration methods wherein a single input port could be selected for use as a bypass to multiple output port subject to restrictions (e.g., output VC must be the same size as the input, different physical link sizes involve bypass links with matching VCs).

BRIEF DESCRIPTION OF THE DRAWINGS

[0036] FIGS. **1**(*a*), **1**(*b*), **1**(*c*) and **1**(*d*) illustrate examples of Bidirectional ring, 2D Mesh, 2D Taurus, and 3D Mesh NoC Topologies.

[0037] FIG. **2**(*a*) illustrates an example of XY routing in a related art two dimensional mesh.

[0038] FIG. **2**(*b*) illustrates three different routes between a source and destination nodes.

[0039] FIG. **3**(*a*) illustrates an example of a related art two layer NoC interconnect.

[0040] FIG. **3**(*b*) illustrates the related art bridge logic between host and multiple NoC layers.

[0041] FIG. **4** illustrates an example of a router, in accordance with an example implementation.

[0042] FIG. **5** illustrates an example flow diagram for configuring routers during configuration time, in accordance with an example implementation.

[0043] FIG. **6** illustrates a valid-ready architecture in accordance with an example implementation.

[0044] FIG. **7**(*a*) illustrates an example system having a SoC element (master), a SoC element (slave), a NoC bridge and a NoC, in accordance with an example implementation. In the example implementation, the NoC bridges and the NoC elements have four input VCs and four output VCs. A single physical wire proceeds from the SoC element to the bridge, whereupon the signal is fanned out to each NoC element in four output VCs.

[0045] FIG. **7**(*b*) illustrates an example architecture for a NoC element, in accordance with an example implementation.

[0046] FIG. **8** illustrates an example table view of information utilized by the NoC element, in accordance with an example implementation.

[0047] FIG. **9** illustrates a flow diagram for a requesting NoC element, in accordance with an example implementation.

DETAILED DESCRIPTION

[0048] The following detailed description provides further details of the figures and example implementations of the present application. Reference numerals and descriptions of redundant elements between figures are omitted for clarity. Terms used throughout the description are provided as examples and are not intended to be limiting. For example, the use of the term "automatic" may involve fully automatic or semi-automatic implementations involving user or administrator control over certain aspects of the implementation, depending on the desired implementation of one of ordinary skill in the art practicing implementations of the present application.

[0049] In example implementations, a NoC interconnect is generated from a specification by utilizing design tools. The specification can contain constraints such as bandwidth/Quality of Service (QoS)/latency attributes that is to be met by the NoC, and can be in various software formats depending on the design tools utilized. Once the NoC is generated through the use of design tools on the specification to meet the specification requirements, the physical architecture can be implemented either by manufacturing a chip layout to facilitate the NoC or by generation of a register transfer level (RTL) for execution on a chip to emulate the generated NoC, depending on the desired implementation.

[0050] In a NoC, there is a network having routers and bridges. Other elements may also be present which can make the NoC fairly large. There may be an inherent latency problem with the NoC. In example implementations, bridges require activation for send messages into the network, and when messages are sent through the link, the router has to arbitrate the messages before the message is sent to the next hop.

[0051] For each hop running at a slow frequency, an entire router arbitration calculation including the travel time can be determined. However, most related art implementations are executed at a high frequency, wherein in such cases that the router arbitration may be conducted in a single cycle. Further, latency can be incurred in the bridge, with a cycle incurred in the bridge, a cycle for the link, a cycle for the router, and so on for the transaction. Latency reduction can be difficult due to the routers having arbitration requirements which incur a latency loss for arbitration in each router.

[0052] FIG. **4** illustrates an example of a router, in accordance with an example implementation. In example implementations for reducing latency in the router, routers implement a fast path, which functions as a bypass having bypass logic **406**. A router may have an assortment of inputs which are processed by elements such as a decoder **401**, a queue such as a First in First Out (FIFO) queue **402**, an arbiter **403** and a multiplexer **404** (mux) for conducting arbitration and determining the output **405**. In example implementations alongside the multiplexer **404**, the router has a path configured to function as a special bypass with bypass logic **406**. One or more inputs can be designated for the special bypass, such that the input entering one of the muxes will be able to hop in at the end of a cycle. If there is an output, the output can be placed in at the end of a cycle so that the input into the router will be able to go directly to the output instead of going through the arbitration. In such an example implementation, one cycle of latency can thereby be removed per router by reducing the processing to decode, bypass logic (e.g. validation) and output. Routing information can be included in direct wires to the router in accordance with the desired implementation. Further, once latency is reduced, the potential round trip latency is decreased as other messages may be able to pop off the FIFO more quickly. Once the bypasses are configured for each eligible router, the example implementations could then calculate the cycle of depth based on this latency. Example implementations of a NoC contains hardware or NoC elements that involve a plurality of physical links and virtual links, with a configurable bypass between virtual links, and bypass logic **406** configured to bypass the queue and the logic of the NoC element. The bypass logic **406** can be configured to initiate bypass of the message in an opportunistic manner (e.g., depending on whether queue is free or not, etc.)

[0053] In example implementations, messages destined to bypass can be pre-arbitrated and then the only logic in the hop can be for determining which output channel is used for the bypass as determined by the bypass logic as illustrated in FIG. **4**. In example implementations, multiple outputs can be used for bypass for an input. For example, one input can bypass to one of multiple output ports, with each output associated with only one input. Bypass logic may also be utilized for optimizing messages in accordance with an example implementation. For example, if a queue is empty the message is sent through the logic for the bypass. If no other message takes priority then the message is transmitted

through the bypass path to avoid all logic. Such example implementations can therefore be configured to conduct more than simply bypassing the FIFO queue and entering arbitration, but can be utilized to bypass all router logic and go directly to the output. In example implementations, the bypass can be conducted when there is no other traffic going on the link, which indicates no cost to arbitration as determined by the bypass logic.

[0054] In the following example implementations, requirements may be set for forwarding an input to the special bypass. One example requirement is that the link sizes are matched so latency from a width conversion is removed. Another example requirement is no clock crossing, so latency from clock conversion is also removed. Other requirements may also be set in according to the desired implementation.

[0055] Related art implementations implement a bypass path in a fixed position that is affixed to an input that is considered to be the most common bypass user. One example of a related art implementation is that an input destined for a particular direction will continually proceed in the direction (e.g. a south input port bypasses to the north input port). Such related art solutions are static.

[0056] In example implementations as illustrated in FIG. **4**, there can be a NoC hardware element which can involve a plurality of physical channels and virtual channels, and a configurable bypass between virtual links, whereupon bypass logic can be configured to bypass the queue and the logic of the NoC element in an opportunistic manner. The bypass logic can allow messages to be transmitted through the bypass opportunistically based on whether the input First in First Out (FIFO) queue is empty or not, based on the priority of the traffic being arbitrated, whether the bypass is idle/available or not, queue depth of the transmitting hardware element, and so on depending on the desired implementation.

[0057] In example implementations, the bypass configuration can be made during configuration time for the specification. FIG. **5** illustrates an example flow diagram for configuring routers during configuration time, in accordance with an example implementation. At **501**, the specification is processed for traffic flows. During configuration time, the example implementations determine all of the traffic flows from the specification, wherein routers that are eligible for bypass are identified at **502**. In an example implementation, if all the traffic flows can be considered during configuration time, traffic tendencies can be identified for a router (e.g. most traffic for an identified router proceeds from the west port to the north port). In the above example, a bypass can be constructed from the west port to the north port to reduce latency. Other implementations based on the traffic flow for identifying routers are also possible depending on the desired implementation. For example, latency sensitivity of traffic flows can also be recognized. In this manner, example implementations can be configured to determine the bypass not only by the most amount of traffic going through a port, the bypass can be determined based on determining the importance of the traffic. Traffic flows can be associated with a weight in terms of the importance of the latency, e.g. how latency sensitive is the traffic, which can be taken into account for identifying eligible routers. Example implementations can calculate the latency sensitivity based on the weights. For example, latency sensitive traffic can be multiplied by the weight to prioritize latency sensitive traffic over raw latency for a channel, depending on the desired implementation.

[0058] Example implementations can also analyze traffic flows so that an array is created based on the input ports (e.g. A, B, C, D, E, and F), and analyze how much of the traffic is coming in on a given link is going to a given output port. So for a given output port, analysis can be conducted by comparing the input ports and constructing a bypass based on the bandwidth consumed by the input ports to a given output port. For example, for a router wherein input port one is responsible for three gigabytes of output for output port X for a given time frame and input port two is responsible for six gigabytes for the given time frame, a bypass can be utilized between input port two and output port one.

[0059] At **503**, locations for implementing a bypass are identified. The locations for implementing the bypass can be identified based on the traffic flow determinations, the hardware configuration of the router and by other methods according to the desired implementation. For example, simulations can be conducted to detect where latency as affected by wire length and travel length are taken into consideration. In such example implementations, output ports can be configured so that a bypass can be made available within the router. And so by converting the router with additional output ports, latency can be reduced. Thus, in example implementations, the optimization can involve determining which bypasses can be implemented to reduce latency and the location of such bypass. The optimization can involve a pre-optimization implementation where conditions for bypassing are identified, and bypasses can be implemented therein. By using design tools during the configuration time, path input algorithms can be utilized to determine the shortest path for the bypass for use in determining the location for implementing the bypass. Optimizations for placement of network elements can also be made to create additional opportunities for bypass in accordance with the desired implementation.

[0060] Bypasses may also be determined based on desired constraints. In an example constraint, the input VC width is set to match the output VC width. In such an example implementation, the physical link size may be different, however, the bypass is still utilized between the two physical links to connect matching input and output VCs.

[0061] At **504**, the eligible routers are then configured with the bypass based on the determinations. As the routers are configurable in example implementations, a heterogeneous NoC with heterogeneous routers can thereby be implemented. Example implementations are in contrast to related art systems, which are directed to homogenous NoC systems and homogenous routers. Related art implementations involve bypasses that are stacked directionally on the assumption that the NoC is homogenous and is therefore static, whereas the example implementations of the present disclosure can utilize heterogeneous router and NoC configurations.

[0062] Example implementations described herein can be implemented as a hardwired bypass. In such example implementations, the software at configuration time can precompute where packets are going and can also utilize sideband information to the NoC. Sideband channels can be utilized for messages to determine which output port to utilize. Sideband information does not need to be utilized for controlling multiplexing to the output ports, but can be

utilized control the validity of the output port. The routing information is processed, wherein example implementations calculate the route including the port.

[0063] As illustrated in FIG. **5**, example implementations can also involve methods and computer readable mediums with instructions directed to determining the selection of bypasses for NoC construction. Such example implementations can involve algorithms that during NoC construction, create additional opportunities for bypassing. The opportunities can involve the reshaping of NoC topology to create more channels that are eligible for bypass (e.g., building a NoC with routers having equal numbers of ports without any clock crossing), applying restrictions to bypass to avoid channels or virtual channels that conduct upsizing and downsizing, and so on depending on the desired implementation.

[0064] Example implementations can also involve algorithms for the creation of bypass paths. As illustrated in FIG. **5**, such algorithms determine all of the possible bypass opportunities for the configurations based on the restrictions as described above. For each possible bypass, the algorithm can then determine which inputs go to which output based on the calculation of expected traffic flows/bandwidth. Such example implementations will determine which bypass provides the biggest impact on the NoC specification (weighted average of traffic, also take latency and importance of traffic into consideration), whereupon the algorithm can thereby choose bypasses with the biggest benefit above a desired threshold.

[0065] Example implementations may also involve algorithms for selecting which multiplexer to incorporate into the NoC hardware element, which can be conducted in a preselected manner or configured after the NoC is designed, in accordance with the desired implementation.

[0066] Example implementations may also involve NoCs with hardware elements having differing physical channel sizes, but VCs with matching sizes to facilitate the bypass. The hardware elements may also be in the form of a configurable router that has complete configurability in terms of which bypasses are available. In an example implementation, the router design can involve having each output port associated with a selected input port with a direct bypass. Further, example implementations may involve a NoC element and configuration method wherein a single input port could be selected from bypass to multiple with restrictions. (e.g., if the output VC is the same size as the input.)

[0067] Virtualization Interface and Valid-Ready for Virtual Channels (VCs) and Other Types of Traffic

[0068] In related art implementations, NoC systems utilize a valid/ready handshake. In such a handshake protocol, one NoC element asserts a valid signal, and if the receiving NoC element asserts a ready signal at the same time, then a message transfer can occur between the two NoC elements. Such related art implementations may further have restrictions depending on the implementation (e.g. to prevent deadlock). In an example restriction, the NoC element does not wait for the valid signal to assert a ready signal, or vice versa. However, related art implementations of the valid/ready handshake are not aware of the actual status of VCs. In related art implementations, even if a request is made using the valid/ready handshake, the status of the VC to be used may actually be blocked. Further, other VCs within the physical channel may be available, but the related art implementations cannot discern their availability due to the NoC elements requiring a ready signal before proceeding. Such implementations may also apply to other traffic types where the valid/ready handshake is blocking the transmission. The destination element would benefit from being able to indicate which traffic flows it would like to receive through the issuance of credits or indication through the ready signal for that specific traffic type.

[0069] In example implementations, additional information is provided for a valid-ready handshake to address the issues with the related art. Example implementations utilize a valid-ready and credit based hybrid system to facilitate valid-ready handshake functionality. In a credit-based design for the example implementations, independent credits are allocated for each VC. The requesting NoC element transmits a request when a VC credit has been obtained.

[0070] Related art implementations utilize a sideband information channel to indicate which virtual channels are available. However, such information is potentially stale. Further, such implementations provide a bit vector that indicates VCs within a range are available (e.g. VCs 8-16) without specifically indicating which VCs are available and which are not.

[0071] FIG. **6** illustrates a valid-ready architecture in accordance with an example implementation. In example implementations, a hybrid approach involving a credit base system is utilized, which facilitates a bi-directional communication. For a NoC requesting element **601** and a NoC target element **602**, there is a valid-ready handshake as well as another vector for VC valid and VC credit in the sideband. The VC valid information is provided to the NoC requesting element **601**, so that the NoC requesting elements makes the request if a specific resource dedicated to the request is available. Such example implementations provide flexibility as the number of virtual channels can be any number in accordance with a desired implementation.

[0072] In example implementations, a number of VCs on the NoC are associated with a physical interface. The physical interface can be associated with a number of interface VCs which can be mapped according to the desired implementation.

[0073] In an example implementation involving a master and slave, a NoC bridge is utilized. The NoC bridge communicates with a slave, which may have a plurality of virtual channels for the traffic. One virtual channel may involve high-priority CPU traffic (e.g. latency-sensitive traffic), another may involve I/O traffic, and another may involve asynchronous traffic which may be time critical, and so on. The properties of the virtual channels may also change over time, depending on the desired implementation.

[0074] In example implementations involving credit based implementation, as each channel can be separated and dedicated to the desired implementation, such implementations avoid the merger of traffic flows that should not be merged.

[0075] In the example implementation hybrid approach, the credit-based handshake is conducted between the agents while valid-ready requirements are enforced. In an example implementation, the target sends a credit back to the master indicating that a resource is available for a request. When the master tries to make that request, the target can indicate that it is not ready due to some delay (e.g. clock crossing). By utilizing the valid-ready with the credit system, it provides a way for temporary back-pressuring from the slave.

[0076] In example implementations, initialization is also facilitated as when the credit-based approach is applied, the NoC elements will determine the initialization. For example, the initialization of the credits can be zero, whereupon after a reset credits can be passed from the target NoC element to the requesting NoC element. Depending on the desired implementation, a certain number of credits can be provided at the master. However, if the reset for the NoC elements are unknown, the flow is harder to control, the valid-ready handshake can be utilized with the ready allowed for de-assertion. Even though the master element has VC credits, the master may be unable to transmit until the target NoC (slave) element is ready to accept the credits.

[0077] In example implementations, different virtual channels may involve different responses (e.g. read response, write response). In example implementations, there can be multiple virtual channels on the read interface going into another controller having only one read response channel. Thus, the congestion may go to the memory controller undergoing different arbitrations with a guaranteed drain. Each channel is completely independent, and they can be used for any purpose according to the desired implementation.

[0078] Example implementations involve a bookkeeping mechanism to track responses. Such a mechanism can involve a data structure to store information to track responses and when the responses are received. For example, if there are four VCs, the VCs can be broken into four segments with reservations. The arbiter may determine to send a flit if the NoC element has credit at the output. The example implementations can involve any partition of the data structure between the four VCs in any way according to the desired implementation. For example, each hardware element can be dedicated to a single VC, or pools of resources can be shared with some or all of the VCs. In example implementations, a mix of dedicated and shared resources can also be provided. Dedicated resources can ensure one channel cannot block another channel.

[0079] FIG. **7**(*a*) illustrates an example system having a SoC element (master) **701**, a SoC element (slave) **705**, NoC bridges **702**, **704** and a NoC **703**, in accordance with an example implementation. In the example implementation, the NoC bridges **702**, **704** and the NoC elements inside the NoC **703** have four input VCs and four output VCs. A single physical wire proceeds from the SoC element to the bridge, whereupon the signal is fanned out to each NoC element in four output VCs.

[0080] FIG. **7**(*b*) illustrates an example architecture for a NoC element, in accordance with an example implementation. In the example implementation, the NoC element has four input VCs and four output VCs. In the example of FIG. **7**(*b*), there is a decoder **711** for the input VCs, a queue **712**, an arbiter **713**, a multiplexer **714**, and an output **715** facilitating output to four output VCs. The single bus feeds into the decoder **711**, which receives the input and fans out the input to four individual queues **712**. When a VC credit is received, the arbiter **713** pops a flit off of the queue **712** and send the flit to the multiplexer **714** to be transmitted through the corresponding output VC **715**.

[0081] As illustrated in FIGS. **7**(*a*) and **7**(*b*), example implementations may involve a NoC that can involve a plurality of channels (e.g., physical channels, virtual channels and/or virtual channels disposed within the physical channels) and NoC hardware elements. Such NoC hardware elements can involve at least one receiving hardware element (e.g., target NoC element **602**) and at least one transmitting hardware element (e.g., requesting NoC element **601**) as illustrated in FIG. **6**. When a transmitting hardware element is to transmit a message, the protocol as illustrated in FIG. **6** can be followed wherein the hardware element transmits a valid signal to the at least one receiving hardware element on a channel of the plurality of channels, and transmits a virtual channel (VC) valid signal on a virtual channel of the plurality of channels to the at least one receiving hardware element. The receiving hardware element is configured to transmit a VC credit to the at least one transmitting hardware element over the virtual channel of the plurality of channels as illustrated in FIG. **6**.

[0082] Depending on the desired implementation, the transmitting hardware element can be configured to not transmit the VC valid signal on the virtual channel until a VC credit is obtained, and transmit the VC valid signal on the virtual channel to the at least one receiving hardware element on receipt of the VC credit based on the protocol of FIG. **6**. In example implementations, the transmitting hardware element can issue a write request when the transmitter determines that the receiving NoC hardware element has enough buffer size for the address information and the storage of data. The transmitting NoC hardware element can infer such information based on the default storage (e.g., 64B) which can be programmable or definable depending on the desired implementation.

[0083] In an example implementation, the plurality of channels can also involve virtual channels, with each of the physical channels being configurable to be independently controlled to adjust a number of VCs for each of the plurality of channels. Such implementations can be conducted by a NoC controller which is configured to define the number of VCs for a given physical channel. In an example implementation, the NoC may maintain the same quantity of VCs for read messages as for read response messages within a given physical channel through such a NoC controller, or they can be differing quantities depending on the desired implementation.

[0084] In example implementations, the NoC may include a configurable interface for the transmitting hardware element and the receiving hardware element, that configures the transmitting hardware element and the receiving hardware element for at least one of deadlock avoidance and quantity of virtual channels. Such configuration can be conducted through a NoC specification, wherein the interface can be in the form of a hardware/software interface or a hardware mechanism that processes the specification to configure the NoC for deadlock avoidance, and quantity of virtual channels.

[0085] In example implementations, the NoC may also include a virtual interface for virtual channels to interact with agents of a SoC. Such a virtual interface can be implemented in the NoC bridges, or can be part of the NoC depending on the desired implementation.

[0086] In example implementations, the transmitting element can be configured to manage VC credits received from one or more receiving hardware elements as illustrated in FIG. **8**, and conduct arbitration based on whether a message destination is associated with a VC credit from the managed VC credits. The hardware elements can be configured to conduct informed arbitration, as each hardware element

knows whether a potential output VC has an associated credit or not based on the information managed as illustrated in FIG. **8**.

[0087] In further example implementations, the receiving hardware element can be configured to provide a reservation for a VC to one or more transmitting hardware elements based on at least one of management of dedicated VC credits to the one or more of transmitting hardware elements, a shared tool providing certain minimum priority for the one or more transmitting hardware elements, and an inference of priority from the one or more of the at least one transmitting hardware element. Such reservations can include a pre-configuration so that certain hardware elements always have a certain number of VC credits reserved, priority inferred based on the type of message received or a hierarchy of hardware elements as defined in the NoC specification.

[0088] FIG. **8** illustrates an example table view of information utilized by the NoC element, in accordance with an example implementation. In example implementations, NoC elements may include a bookkeeping mechanism to indicate the status of the target VCs. In the example of FIG. **8**, each output VC is associated with a ready signal, and VC credit. When ready and valid are set, then a transfer can take place. VC credit indicates the number of credits available for transmission to the output VC. VC credit is incremented when a credit signal is received, and decremented when a credit is utilized.

[0089] FIG. **9** illustrates a flow diagram for a requesting NoC element, in accordance with an example implementation. At **901** the requesting NoC element waits until a VC credit is received before transmitting a request. At **902**, once a VC credit is received, the requesting NoC element conducts arbitration among available traffic that are associated with credits, and forwards the data packet to the output interface. At **903**, the valid/ready handshake as illustrated in FIG. **6** is conducted, wherein a VC valid signal is provided to indicate the VC that the data will be sent through and the data/flit is sent through the corresponding VC with the VC valid signal. The VC credit counter is decremented. The requesting NoC element will also wait for additional VC credits as necessary. At **904**, the receiving element receives the data/flit from the transmitting element.

[0090] In example implementations there can be a system such as a NoC, a SoC, or any hardware element system that require a virtual channel interface that involves a plurality of channels; at least one receiving hardware element; and at least one transmitting hardware element configured to: transmit a valid signal to the at least one receiving hardware element on a channel of the plurality of channels, and transmit a virtual channel (VC) valid signal as a virtual channel indicator for a virtual channel of a plurality of virtual channels designated for transmission of data and transmit the data on the virtual channel designated for the transmission of the data; wherein the at least one receiving hardware element is configured to transmit a VC credit to the at least one transmitting hardware element as illustrated in FIG. **6** and FIG. **9**.

[0091] In example implementations, the at least one transmitting hardware element is configured to not transmit the data packet on the virtual channel until a VC credit is obtained. The plurality of channels can be physical channels that are partitioned into one or more virtual channels, and each of the channels can be configurable to be independently controlled for mapping to an interface virtual VCs. In such example implementations, multiple transmitting channels can map to a single interface virtual channel, or a single transmitting channel can map to multiple virtual channels depending on the desired implementation. In an example implementation involving a single transmitting channel mapping to multiple virtual channels, the transmission can be conducted when any of the VC credits are available. The mapping can be done through a virtual interface connected to the NoC to map virtual channels with transmitting elements such as agents of a SoC. Such interfaces can include read channels, read response channels, and so on depending on the desired implementation. In example implementations, the interface can include the decoder, queue, arbiter, multiplexer, and/or the output as illustrated in FIG. **7**(*b*).

[0092] In example implementations, the at least one transmitting element is further configured to manage VC credits received from one or more of the at least one receiving hardware element; and conduct arbitration based on whether a message destination is associated with a VC credit from the managed VC credits as illustrated in FIG. **8**. The management can be done through the interface of the hardware element that is configured to map transmitting channels to virtual channels.

[0093] In example implementations, the at least one transmitting hardware element is configured to arbitrate messages for transmitting through prioritizing messages that are associated with a VC credit through the user of the arbiter as illustrated in FIG. **7**(*b*).

[0094] In example implementations, the at least one receiving hardware element is configured to provide a reservation for a VC to one or more of the at least one transmitting hardware element based on at least one of management of dedicated VC credits to the one or more of the at least one transmitting hardware element, and an inference of priority from the one or more of the at least one transmitting hardware element based on the information of FIG. **8**. Priority can be inferred based on the type of message and the hierarchy set according to the desired implementation (e.g., hierarchy for read, read response, write, etc.).

[0095] In example implementations the at least one receiving hardware element can be a NoC element such as a router or a bridge and the at least one transmitting hardware element is an agent of the System on Chip (SoC), such as a memory or a CPU.

[0096] Although example implementations involve a NoC, other systems such as a SoC or other interconnect can be utilized in accordance with the desired implementation. Any hardware element that can utilize a virtual interface can take advantage of the example implementations described herein.

[0097] Unless specifically stated otherwise, as apparent from the discussion, it is appreciated that throughout the description, discussions utilizing terms such as "processing," "computing," "calculating," "determining," "displaying," or the like, can include the actions and processes of a computer system or other information processing device that manipulates and transforms data represented as physical (electronic) quantities within the computer system's registers and memories into other data similarly represented as physical quantities within the computer system's memories or registers or other information storage, transmission or display devices.

[0098] Example implementations may also relate to an apparatus for performing the operations herein. This appa-

ratus may be specially constructed for the required purposes, or it may include one or more general-purpose computers selectively activated or reconfigured by one or more computer programs. Such computer programs may be stored in a computer readable medium, such as a computer-readable storage medium or a computer-readable signal medium. A computer-readable storage medium may involve tangible mediums such as, but not limited to optical disks, magnetic disks, read-only memories, random access memories, solid state devices and drives, or any other types of tangible or non-transitory media suitable for storing electronic information. A computer readable signal medium may include mediums such as carrier waves. The algorithms and displays presented herein are not inherently related to any particular computer or other apparatus. Computer programs can involve pure software implementations that involve instructions that perform the operations of the desired implementation.

[0099] Various general-purpose systems may be used with programs and modules in accordance with the examples herein, or it may prove convenient to construct a more specialized apparatus to perform desired method steps. In addition, the example implementations are not described with reference to any particular programming language. It will be appreciated that a variety of programming languages may be used to implement the teachings of the example implementations as described herein. The instructions of the programming language(s) may be executed by one or more processing devices, e.g., central processing units (CPUs), processors, or controllers.

[0100] As is known in the art, the operations described above can be performed by hardware, software, or some combination of software and hardware. Various aspects of the example implementations may be implemented using circuits and logic devices (hardware), while other aspects may be implemented using instructions stored on a machine-readable medium (software), which if executed by a processor, would cause the processor to perform a method to carry out implementations of the present disclosure. Further, some example implementations of the present disclosure may be performed solely in hardware, whereas other example implementations may be performed solely in software. Moreover, the various functions described can be performed in a single unit, or can be spread across a number of components in any number of ways. When performed by software, the methods may be executed by a processor, such as a general purpose computer, based on instructions stored on a computer-readable medium. If desired, the instructions can be stored on the medium in a compressed and/or encrypted format.

[0101] Moreover, other implementations of the present disclosure will be apparent to those skilled in the art from consideration of the specification and practice of the teachings of the present disclosure. Various aspects and/or components of the described example implementations may be used singly or in any combination. It is intended that the specification and example implementations be considered as examples only, with the true scope and spirit of the present disclosure being indicated by the following claims.

What is claimed is:

1. A hardware element incorporated into a Network on Chip (NoC), comprising:

a plurality of physical links and virtual links;

a queue for transmission of output messages to output ports of the hardware element;

an arbiter configured to process input messages to the queue based on a logic scheme;

a configurable bypass link between the virtual links, and

bypass logic configured to redirect the input messages to the configurable bypass link to bypass the queue and the arbiter.

2. The hardware element of claim **1**, wherein the bypass logic is configured to redirect the input messages to the configurable bypass link opportunistically.

3. A hardware element incorporated into a System on Chip (SoC), comprising:

a plurality of physical links and virtual links;

a queue for transmission of output messages to output ports of the hardware element;

an arbiter configured to process input messages to the queue based on a logic scheme;

a configurable bypass link between the virtual links, and

bypass logic configured to redirect the input messages to the configurable bypass link to bypass the queue and the arbiter.

4. The hardware element of claim **3**, wherein the bypass logic is configured to redirect the input messages to the configurable bypass link opportunistically.

* * * * *