

Exhibit F-6

Invalidity of U.S. Patent No. 8,086,800 in view of Radulescu I

Radulescu et al., *Communications Services for Networks on Chip* (“Radulescu I”), which published no later than November 2003, qualifies as prior art under at least 35 U.S.C. §§ 102 and 103 as to the asserted claim of U.S. Patent No. 8,086,800 (“’800 patent”), as discussed in greater detail in the chart that follows.¹

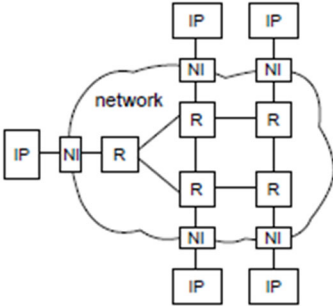
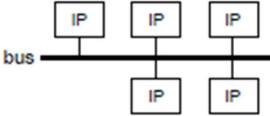
To the extent Plaintiff alleges that Radulescu I does not disclose any particular limitation of claim 12 (the “Asserted Claim”) of the ’800 patent, either expressly or inherently, it would have been obvious to a person of ordinary skill in the art as of the priority date of the ’800 patent to modify Radulescu I and/or to combine the teachings of Radulescu I with other prior art references, including but not limited to the prior art references presently found in Defendants’ Invalidity Contentions and any supplements thereto and the relevant section of charts for other prior art for the ’800 patent in a manner that would have rendered the Asserted Claim invalid as obvious. The inclusion of the particular limitation into the methods and systems described in Radulescu I would have been obvious for at least the following reasons: It would have been obvious to modify the methods and/or systems disclosed in Radulescu I, alone or in combination with the references identified in Defendants’ Invalidity Contentions, to meet this limitation based on: (a) the teachings in Radulescu I itself; (b) a person of ordinary skill in the art implementing a predictable variation of the methods and/or systems disclosed in Radulescu I or in the other identified references; (c) the logic, judgment, intelligence, creativity, innovation and common sense a person of ordinary skill in the art would employ; (d) the inferences and creative and routine steps that a person of ordinary skill in the art would employ; (e) market demand for methods and/or systems to include this limitation; and/or (f) design needs or market pressures to solve a problem with a finite number of solutions that could have produced the desired outcome.

Defendants reserve the right to amend or supplement this claim chart at a later date pursuant to the Docket Control Order issued by the Court on June 16, 2023 (Dkt. No. 59).

¹ Nothing in these claim charts should be construed as an admission regarding infringement, either literally or under the doctrine of equivalents, or as an admission regarding Defendants’ understanding of the proper scope of the Asserted Claim. Given Plaintiff’s vague and ambiguous infringement contentions, the exemplary citations herein necessarily account for a variety of possible infringement arguments and claim constructions. Defendants reserve the right to rely on additional citations or sources of evidence that also may be applicable, or that may become applicable in light of claim construction, changes in Plaintiff’s infringement contentions, and/or information obtained during discovery as the case progresses. Various positions put forth in this document are predicated on Plaintiff’s overly broad interpretation of its claims as evidenced by the infringement allegations received to date. Those positions are not intended to and do not necessarily reflect Defendants’ interpretation of the true and proper scope of Plaintiff’s claims, and Defendants reserve the right to adopt claim construction positions that differ from or even conflict with various positions put forth in this document.

A. Claim 12

'800 Patent Claim Limitation	Radulescu I
[12.pre] An integrated circuit comprising:	To the extent the preamble is limiting, Radulescu I discloses and/or renders obvious the preamble based on at least the following citations. See, e.g.: <i>See, e.g.</i>, p. 275: “Networks are emerging as a possible solution for future on-chip interconnects. In this chapter, we show how networks on chip (NoC) are similar to and differ from both off-chip networks (e.g., computer networks) and current on-chip interconnects (e.g., buses). We re-examine their communication services in the context of NoCs. To enable a clean separation between the NoC and IP blocks, we provide services that abstract from network implementations. We define a request-response transaction model similar to bus protocols, to make our approach backward compatible. To exploit the full power of NoCs, we also provide connection-oriented communication with differentiated services. Examples are bandwidth guarantees, transaction orderings, and end-to-end flow control.” <i>See, e.g.</i>, p. 277: “II. Networks Brought on Chip Networks have been the subject of research for decades, both in the context of local and wide area networks (computer networks) [25], and as an interconnect for parallel machines [6]. Both are very much related to on-chip networks, and many of the results in those fields are also applicable on chip. However, NoC premises are different from off-chip networks, and, therefore, most of the network design choices must be re-evaluated. NoCs differ from off-chip networks mainly in their constraints and synchronization. Typically, resource constraints are tighter on chip than off chip. Storage (i.e., memory) and computation resources are relatively more expensive, whereas the number of point-to-point links is larger on chip than off chip [7].” <i>See, e.g.</i>, p. 280-81: “Introducing networks (Figure 1) as on-chip interconnects radically changes the communication as compared to direct interconnects such as buses or switches (Figure 2). This is because of the multi-hop nature of a network, where communication modules are not directly connected, but separated by one or more network nodes. This is in contrast with the prevalent existing interconnects (i.e., buses) where modules are directly connected. The implications of this

'800 Patent Claim Limitation	Radulescu I
	change reside in the arbitration (which must change from centralized to distributed), and in the communication properties (e.g., ordering, or flow control). In this section, we list some of these topics, and outline the differences between NoCs and buses. We refer mainly to buses as direct interconnects, because currently they are the most used on-chip interconnect. Most of the bus characteristics also hold for other direct interconnects (e.g., switches [16]). Multilevel buses are a hybrid between buses and NoCs. For our purposes, depending on the functionality of the bridges, multilevel buses behave either like simple buses [2] or like NoCs.”  <i>Figure 1. A network interconnect example</i>  <i>Figure 2. A bus interconnect example</i> See, e.g., p. 281: “Programming Model. The programming model of a bus typically consists of load and store operations which are implemented as a sequence of primitive bus transactions. Bus interfaces typically have dedicated groups of wires for command, address, write data, and read data [1, 12, 13, 17, 18, 26]. A bus is a resource shared by multiple IPs. Therefore, before using it, IPs must go through an arbitration phase, where they request access to the bus, and block until the bus is granted to them. A bus transaction involves a request and possibly a response. Modules issuing requests are called masters, and those serving requests are called slaves. If there is a single arbitration for a request-response pair, the bus is called non-split. In this case, the bus remains allocated to the master of the transaction until the response is delivered, even when this takes a long time. Alternatively, in

'800 Patent Claim Limitation	Radulescu I
	a split bus, the bus is released after the request to allow transactions from different masters to be initiated. However, a new arbitration must be performed for the response such that the slave can access the bus [11]. For both split and non-split buses, both communication parties have direct and immediate access to the status of the transaction. In contrast, network transactions are one-way transfers from an output buffer at the source to an input buffer at the destination that causes some action at the destination, the occurrence of which is not visible at the source [6]. The effects of a network transaction are observable only through additional transactions. A request-response type of operation is still possible, but requires at least two distinct network transactions. Thus, a bus-like transaction in a NoC will essentially be a split transaction.” <i>See, e.g., p. 287-88: “IPs interact with our network [8, 20, 21] at so-called network interfaces (NI). NIs provide NI ports (NIP) through which the communication services are accessed. As shown in Figure 3, a NI can have several NIPs to which one or more IPs (computation elements or memories, but not interconnection elements) can be connected. Similarly, an IP can be connected to more than one NI and NIP.”</i> <pre> graph TD IP1[IP] --- Dot1(()) IP2[IP] --- Dot2(()) Dot1 --- NI1[NI] Dot2 --- NI1 Dot2 --- NI2[NI] NI1 --- IP1 NI1 --- IP2 NI2 --- IP1 </pre> Figure 3. Examples of links between NIs and IPs. <i>See, e.g., p. 288: “Communication between NIPs is performed on connections. Connections are introduced to describe and identify communication with different properties, such as guaranteed throughput, bounded latency and jitter, ordered delivery, or flow control. For example, to distinguish and independently guarantee communication of 1Mbps and 25Mbps, two connections can be used. Two NIPs can be connected by multiple connections, possibly with different properties. Connections as defined here are similar to the concept of threads and connections from OCP and VCI. Where in OCP and VCI connections are used only to relax transaction ordering, we generalize</i>

'800 Patent Claim Limitation	Radulescu I
	from only the ordering property to include configuration of buffering and flow control, guaranteed throughput, and bounded latency per connection.” <i>See, e.g.</i>, p. 288: “To allow more flexibility in configuring connections and, hence, better resource allocation per connection, the outgoing and return parts of connections are configured independently. For example, a different amount of buffer space can be allocated in the NIPs at master and slaves, or different bandwidths can be reserved for requests and responses.” <i>See, e.g.</i>, p. 297: “We also compare NoCs to existing on-chip interconnects, such as buses and switches. By directly connecting IP blocks, existing on-chip interconnects can offer tight coupling between masters and slaves, and global arbitration. In NoCs, masters and slaves are completely decoupled, and the arbitration is distributed over the network nodes. This make it harder to provide guarantees, such as bandwidth lower bounds, and transaction orderings.” To the extent Radulescu I does not expressly disclose the preamble, a person of ordinary skill in the art would have determined that the preamble is either inherent and/or obvious to one of ordinary skill in the art in view of the teachings of Radulescu I. Further, one of ordinary skill in the art would have been motivated to modify Radulescu I or combine it with any of the present prior art references found in Defendant’s Invalidity Contentions and any supplements thereto and the relevant section charts for other prior art for the ’800 Patent in a manner that would result in the subject matter of the preamble given, at the very least, the person’s understanding of the state of the art, the problems addressed and solved in the prior art, and the teachings of Radulescu I.
[12.a] a plurality of processing modules including a first processing module having a first processing memory and a second processing module having a second processing memory;	Radulescu I discloses and/or renders obvious this limitation based on at least the following citations. <i>See, e.g.</i>: <i>See, e.g.</i>, p. 275: “Networks are emerging as a possible solution for future on-chip interconnects. In this chapter, we show how networks on chip (NoC) are similar to and differ from both off-chip networks (e.g., computer networks) and current on-chip interconnects (e.g., buses). We re-examine their communication services in the context of NoCs. To enable a clean separation between the NoC and IP blocks, we provide services that abstract from network implementations. We define a

'800 Patent Claim Limitation	Radulescu I
	request-response transaction model similar to bus protocols, to make our approach backward compatible. To exploit the full power of NoCs, we also provide connection-oriented communication with differentiated services. Examples are bandwidth guarantees, transaction orderings, and end-to-end flow control.” <i>See, e.g.</i>, p. 278: “In the rest of the section, we list five network issues that have a direct impact on the NoC cost: reliable communication, deadlock, data ordering, network flow control and buffering strategy, and time-related guarantees. For each of them, we discuss the differences and similarities for on- and off-chip networks.” <i>See, e.g.</i>, p. 280-81: “Introducing networks (Figure 1) as on-chip interconnects radically changes the communication as compared to direct interconnects such as buses or switches (Figure 2). This is because of the multi-hop nature of a network, where communication modules are not directly connected, but separated by one or more network nodes. This is in contrast with the prevalent existing interconnects (i.e., buses) where modules are directly connected. The implications of this change reside in the arbitration (which must change from centralized to distributed), and in the communication properties (e.g., ordering, or flow control). In this section, we list some of these topics, and outline the differences between NoCs and buses. We refer mainly to buses as direct interconnects, because currently they are the most used on-chip interconnect. Most of the bus characteristics also hold for other direct interconnects (e.g., switches [16]). Multilevel buses are a hybrid between buses and NoCs. For our purposes, depending on the functionality of the bridges, multilevel buses behave either like simple buses [2] or like NoCs.”

'800 Patent Claim Limitation	Radulescu I
	<i>Figure 1. A network interconnect example</i> <i>Figure 2. A bus interconnect example</i> <i>See, e.g., p. 281: “Programming Model. The programming model of a bus typically consists of load and store operations which are implemented as a sequence of primitive bus transactions. Bus interfaces typically have dedicated groups of wires for command, address, write data, and read data [1, 12, 13, 17, 18, 26].</i> A bus is a resource shared by multiple IPs. Therefore, before using it, IPs must go through an arbitration phase, where they request access to the bus, and block until the bus is granted to them. A bus transaction involves a request and possibly a response. Modules issuing requests are called masters, and those serving requests are called slaves. If there is a single arbitration for a request-response pair, the bus is called non-split. In this case, the bus remains allocated to the master of the transaction until the response is delivered, even when this takes a long time. Alternatively, in a split bus, the bus is released after the request to allow transactions from different masters to be initiated. However, a new arbitration must be performed for the response such that the slave can access the bus [11]. For both split and non-split buses, both communication parties have direct and immediate access to the status of the transaction. In contrast, network transactions are one-way transfers from an output buffer at the source to an input buffer at the destination that causes some action at the destination, the occurrence of which is not visible at the source [6]. The effects of a network transaction are observable only through additional transactions. A request-response type of

'800 Patent Claim Limitation	Radulescu I
	operation is still possible, but requires at least two distinct network transactions. Thus, a bus-like transaction in a NoC will essentially be a split transaction.” <i>See, e.g.,</i> p. 285: “Buffering and Flow Control. Buffering data of a master (output buffering) is used both for buses and NoCs to decouple computation from communication. However, for NoCs output buffering is also needed to marshal data, which consists of (a) (optionally) splitting the outgoing data in smaller packets which are transported by the network, and (b) adding control information for the network around the data (packet header). To avoid output buffer overflow the master must not initiate transactions that generate more data than the currently available space. Similarly to output buffering, input buffering is also used to decouple computation from communication. In a NoC, input buffering is also required to unmarshal data. In addition, flow control for input buffers differs for buses and NoCs. For buses, the source and destination are directly linked, and, destination can therefore signal directly to a source that it cannot accept data. This information can even be available to the arbiter such that the bus is not granted to a transaction trying to write to a full buffer. In a NoC, however, the destination of a transaction cannot signal directly to a source that its input buffer is full. Consequently, transactions to a destination can be started, possibly from multiple sources, after the destination’s input buffer has filled up. Several policies can be adopted when an input buffer is full. One policy is not to accept additional incoming transitions, and to store them in the network. However, this approach can easily lead to network congestion, as the data could be eventually stored all the way to the sources, blocking the links in between. Another policy is to accept incoming transactions at a full destination, and drop some data in the input buffer. Congestion is avoided but data is lost, and this is undesirable. To avoid input buffer overflow connections can be used, together with end-to-end flow control. At connection set up between a master and one or more slaves, buffer space is allocated at the network interfaces of the slaves, and the network interface of the master is assigned credits reflecting the amount of buffer space at the slaves. The master can only send data when it has enough credits for the destination slave(s). The slaves grant credits to the master when they consume data.”

'800 Patent Claim Limitation	Radulescu I
	<i>See, e.g.</i>, p. 287-88: “IPs interact with our network [8, 20, 21] at so-called network interfaces (NI). NIs provide NI ports (NIP) through which the communication services are accessed. As shown in Figure 3, a NI can have several NIPs to which one or more IPs (computation elements or memories, but not interconnection elements) can be connected. Similarly, an IP can be connected to more than one NI and NIP.” Figure 3. Examples of links between NIs and IPs. <i>See, e.g.</i>, p. 288: “Communication between NIPs is performed on connections. Connections are introduced to describe and identify communication with different properties, such as guaranteed throughput, bounded latency and jitter, ordered delivery, or flow control. For example, to distinguish and independently guarantee communication of 1Mbps and 25Mbps, two connections can be used. Two NIPs can be connected by multiple connections, possibly with different properties. Connections as defined here are similar to the concept of threads and connections from OCP and VCI. Where in OCP and VCI connections are used only to relax transaction ordering, we generalize from only the ordering property to include configuration of buffering and flow control, guaranteed throughput, and bounded latency per connection.” <i>See, e.g.</i>, p. 288: “To allow more flexibility in configuring connections and, hence, better resource allocation per connection, the outgoing and return parts of connections are configured independently. For example, a different amount of buffer space can be allocated in the NIPs at master and slaves, or different bandwidths can be reserved for requests and responses.” <i>See, e.g.</i>, p. 289: “Communication takes place on connections using transactions, consisting of a request and possibly a response. The request encodes an operation (e.g., read, write, flush, test and set, nop) and possibly carries outgoing data (e.g., for write commands). The response returns data as a result of a command (e.g., read) and/or an acknowledgment.”

'800 Patent Claim Limitation	Radulescu I
	<i>See, e.g., p. 289:</i> “Connections involve at least two NIPs. Transactions on a connection are always started at one and only one of the NIPs, called the connection’s active NIP (ANIP). All the other NIPs of the connection are called passive NIPs (PNIP).” <i>See, e.g., p. 289:</i> “There can be multiple transactions active on a connection at a time, but more generally than for split buses. That is, transactions can be started at the ANIP of a connection while responses for earlier transactions are pending. If a connection has multiple slaves, multiple transactions can be initiated towards different slaves. Transactions are also pipelined between a single master-slave pair for both requests and responses. In principle, transactions can also be pipelined within a slave, if the slave allows this.” <i>See, e.g., p. 289-91:</i> “A transaction is composed from the following messages (see Figure 4): • A command message (CMD) is sent by the ANIP, and describes the action to be executed at the slave connected to the PNIP. Examples of commands are read, write, test and set, and flush. Commands are the only messages that are compulsory in a transaction. For NIPs that allow only a single command with no parameters (e.g., fixed-size address-less write), we assume the command message still exists, even if it is implicit (i.e., not explicitly sent by the IP). • An out data message (OUTDATA) is sent by the ANIP following a command that requires data to be executed (e.g., write, multicast, and test-and-set). • A return data message (RETDATA) is sent by a PNIP as a consequence of a transaction execution that produces data (e.g., read, and test-and-set). • A completion acknowledgment message (RETSTAT) is an optional message which is returned by PNIP when a command has been completed. It may signal either a successful completion or an error. For transactions including both RETDATA and RETSTAT the two messages can be combined in a single message for efficiency. However, conceptually, they exist both: RETSTAT to signal the presence of data or an error, and RETDATA to carry the data. In bus-based interfaces RETDATA and RETSTAT typically exist as two separate signals [1, 12, 13, 17, 18, 26].

'800 Patent Claim Limitation	Radulescu I
	Communication takes place on connections using transactions, consisting of a request and possibly a response. The request encodes an operation (e.g., read, write, flush, test and set, nop) and possibly carries outgoing data (e.g., for write commands). The response returns data as a result of a command (e.g., read) and/or an acknowledgment. Connections involve at least two NIPs. Transactions on a connection are always started at one and only one of the NIPs, called the connection's active NIP (ANIP). All the other NIPs of the connection are called passive NIPs (PNIP). There can be multiple transactions active on a connection at a time, but more generally than for split buses. That is, transactions can be started at the ANIP of a connection while responses for earlier transactions are pending. If a connection has multiple slaves, multiple transactions can be initiated towards different slaves. Transactions are also pipelined between a single master-slave pair for both requests and responses. In principle, transactions can also be pipelined within a slave, if the slave allows this. A transaction is composed from the following messages (see Figure 4): • A command message (CMD) is sent by the ANIP, and describes the action to be executed at the slave connected to the PNIP. Examples of commands are read, write, test and set, and flush. Commands are the only messages that are compulsory in a transaction. For NIPs that allow only a single command with no parameters (e.g., fixed-size address-less write), we assume the command message still exists, even if it is implicit (i.e., not explicitly sent by the IP). • An out data message (OUTDATA) is sent by the ANIP following a command that requires data to be executed (e.g., write, multicast, and test-and-set). • A return data message (RETDATA) is sent by a PNIP as a consequence of a transaction execution that produces data (e.g., read, and test-and-set). • A completion acknowledgment message (RETSTAT) is an optional message which is returned by PNIP when a command has been completed. It may signal either a successful completion or an error. For transactions including both RETDATA and RETSTAT the two messages can be combined in a single message for efficiency. However, conceptually, they exist both: RETSTAT to signal the presence of data or an error, and RETDATA to carry the

'800 Patent Claim Limitation	Radulescu I
	data. In bus-based interfaces RETDATA and RETSTAT typically exist as two separate signals [1, 12, 13, 17, 18, 26]. Messages composing a transaction are divided in outgoing messages, namely CMD and OUTDATA, and response messages, namely RETDATA, RETSTAT. Within a transaction, CMD precedes all other messages, and RETDATA precedes RETSTAT if present. These rules apply both between master and ANIP, and PNIP and slave. Examples of transactions are shown in Figure 5. We classify connections as follows (see Figure 6): • A simple connection is a connection between one ANIP and one PNIP. • A narrowcast connection is a connection between one ANIP and one or more PNIPs, in which each transaction that the ANIP initiates is executed by exactly one PNIP. An example of the narrowcast connection is shown in Figure 7, where the ANIP performs transactions on an address space which is mapped on two memory modules. Depending on the transaction address, a transaction is executed on only one of these two memories. • A multicast connection is a connection between one ANIP and one or more PNIPs, in which the sent messages are duplicated and each PNIP receives a copy of those messages. In a multicast connection no return messages are currently allowed, because of the large traffic they generate (i.e., one response per destination). It could also increase the complexity in the ANIP because individual responses from PNIPs must be merged into a single response for the ANIP. This requires buffer space and/or additional computation for the merging itself.” <pre> graph LR ANIP[ANIP] -- OUTDATA --> PNIP[PNIP] PNIP -- RETDATA --> ANIP ANIP -- CMD --> PNIP PNIP -- RETSTAT --> ANIP </pre> <i>Figure 4.</i> Transaction composition.

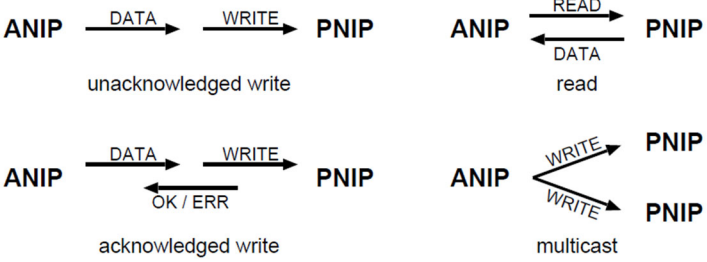
'800 Patent Claim Limitation	Radulescu I
	ANIP $\xrightarrow{\text{DATA}}$ $\xrightarrow{\text{WRITE}}$ PNIP ANIP $\xrightarrow{\text{READ}}$ PNIP unacknowledged write DATA read ANIP $\xrightarrow{\text{DATA}}$ $\xrightarrow{\text{WRITE}}$ PNIP ANIP $\xrightarrow{\text{WRITE}}$ PNIP $\xleftarrow{\text{OK / ERR}}$ $\xrightarrow{\text{WRITE}}$ PNIP acknowledged write multicast <i>Figure 5.</i> Transaction examples. simple: ANIP $\longleftrightarrow$ PNIP narrowcast: ANIP $\longleftrightarrow$ PNIP PNIP PNIP multicast: ANIP $\longleftrightarrow$ PNIP PNIP PNIP <i>Figure 6.</i> Connection types. <i>Figure 7.</i> A narrowcast connection. See, e.g., p. 292: “There are several points in a connection where order of transactions can be observed (see Figure 8): (a) the order in which the master presents CMD messages to the ANIP, (b) the order in which the CMDs are delivered to the slave by the PNIP, (c) the order in which the slave presents the responses to the PNIP, and (d) the order the responses are delivered to the master by the ANIP. Note that not all of (b), (c), and (d) are always present. Moreover, there are no assumptions about the order in which the slaves execute transactions; we can only observe the order of the

'800 Patent Claim Limitation	Radulescu I
	responses. We consider the order of the transaction execution by the slaves to be a system decision, and not a part of the interconnect protocol.” <i>See, e.g.</i>, p. 297: “We also compare NoCs to existing on-chip interconnects, such as buses and switches. By directly connecting IP blocks, existing on-chip interconnects can offer tight coupling between masters and slaves, and global arbitration. In NoCs, masters and slaves are completely decoupled, and the arbitration is distributed over the network nodes. This make it harder to provide guarantees, such as bandwidth lower bounds, and transaction orderings.” To the extent Radulescu I does not expressly disclose this limitation, a person of ordinary skill in the art would have determined that this limitation is either inherent and/or obvious to one of ordinary skill in the art in view of the teachings of Radulescu I. Further, one of ordinary skill in the art would have been motivated to modify Radulescu I or combine it with any of the present prior art references found in Defendant’s Invalidity Contentions and any supplements thereto and the relevant section charts for other prior art for the ’800 Patent in a manner that would result in the subject matter of this limitation given, at the very least, the person’s understanding of the state of the art, the problems addressed and solved in the prior art, and the teachings of Radulescu I.
[12.b] a plurality of interconnect modules including a first interconnect module and a second interconnect module, wherein the first processing module is connected to the first interconnect module and the second processing module is connected to the second interconnect module; and	Radulescu I discloses and/or renders obvious the preamble based on at least the following citations. <i>See, e.g.</i>: <i>See, e.g.</i>, p. 280-81: “Introducing networks (Figure 1) as on-chip interconnects radically changes the communication as compared to direct interconnects such as buses or switches (Figure 2). This is because of the multi-hop nature of a network, where communication modules are not directly connected, but separated by one or more network nodes. This is in contrast with the prevalent existing interconnects (i.e., buses) where modules are directly connected. The implications of this change reside in the arbitration (which must change from centralized to distributed), and in the communication properties (e.g., ordering, or flow control). In this section, we list some of these topics, and outline the differences between NoCs and buses. We refer mainly to buses as direct interconnects, because currently they are the most used on-chip interconnect. Most of the bus characteristics also hold for other direct interconnects (e.g., switches [16]). Multilevel buses are a hybrid between buses and NoCs. For our purposes, depending

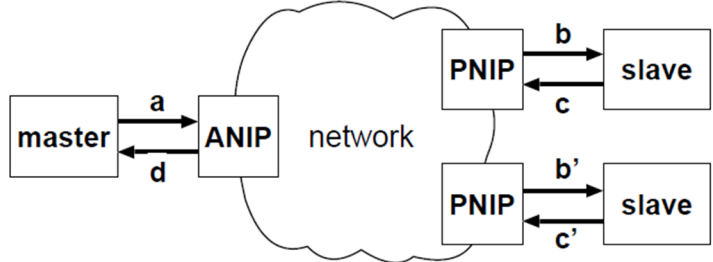
'800 Patent Claim Limitation	Radulescu I
	on the functionality of the bridges, multilevel buses behave either like simple buses [2] or like NoCs.” <i>Figure 1. A network interconnect example</i> <i>Figure 2. A bus interconnect example</i> See, e.g., p. 287-88: “IPs interact with our network [8, 20, 21] at so-called network interfaces (NI). NIs provide NI ports (NIP) through which the communication services are accessed. As shown in Figure 3, a NI can have several NIPs to which one or more IPs (computation elements or memories, but not interconnection elements) can be connected. Similarly, an IP can be connected to more than one NI and NIP.” <i>Figure 3. Examples of links between NIs and IPs.</i> See, e.g., p. 289-91: “A transaction is composed from the following messages (see Figure 4): • A command message (CMD) is sent by the ANIP, and describes the action to be executed at the slave connected to the PNIP. Examples of commands are read, write, test and set, and

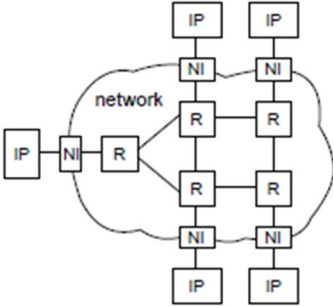
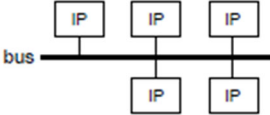
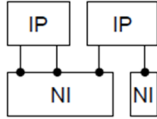
'800 Patent Claim Limitation	Radulescu I
	flush. Commands are the only messages that are compulsory in a transaction. For NIPs that allow only a single command with no parameters (e.g., fixed-size address-less write), we assume the command message still exists, even if it is implicit (i.e., not explicitly sent by the IP). • An out data message (OUTDATA) is sent by the ANIP following a command that requires data to be executed (e.g., write, multicast, and test-and-set). • A return data message (RETDATA) is sent by a PNIP as a consequence of a transaction execution that produces data (e.g., read, and test-and-set). • A completion acknowledgment message (RETSTAT) is an optional message which is returned by PNIP when a command has been completed. It may signal either a successful completion or an error. For transactions including both RETDATA and RETSTAT the two messages can be combined in a single message for efficiency. However, conceptually, they exist both: RETSTAT to signal the presence of data or an error, and RETDATA to carry the data. In bus-based interfaces RETDATA and RETSTAT typically exist as two separate signals [1, 12, 13, 17, 18, 26]. Communication takes place on connections using transactions, consisting of a request and possibly a response. The request encodes an operation (e.g., read, write, flush, test and set, nop) and possibly carries outgoing data (e.g., for write commands). The response returns data as a result of a command (e.g., read) and/or an acknowledgment. Connections involve at least two NIPs. Transactions on a connection are always started at one and only one of the NIPs, called the connection's active NIP (ANIP). All the other NIPs of the connection are called passive NIPs (PNIP). There can be multiple transactions active on a connection at a time, but more generally than for split buses. That is, transactions can be started at the ANIP of a connection while responses for earlier transactions are pending. If a connection has multiple slaves, multiple transactions can be initiated towards different slaves. Transactions are also pipelined between a single master-slave pair for both requests and responses. In principle, transactions can also be pipelined within a slave, if the slave allows this. A transaction is composed from the following messages (see Figure 4):

'800 Patent Claim Limitation	Radulescu I
	• A command message (CMD) is sent by the ANIP, and describes the action to be executed at the slave connected to the PNIP. Examples of commands are read, write, test and set, and flush. Commands are the only messages that are compulsory in a transaction. For NIPs that allow only a single command with no parameters (e.g., fixed-size address-less write), we assume the command message still exists, even if it is implicit (i.e., not explicitly sent by the IP). • An out data message (OUTDATA) is sent by the ANIP following a command that requires data to be executed (e.g., write, multicast, and test-and-set). • A return data message (RETDATA) is sent by a PNIP as a consequence of a transaction execution that produces data (e.g., read, and test-and-set). • A completion acknowledgment message (RETSTAT) is an optional message which is returned by PNIP when a command has been completed. It may signal either a successful completion or an error. For transactions including both RETDATA and RETSTAT the two messages can be combined in a single message for efficiency. However, conceptually, they exist both: RETSTAT to signal the presence of data or an error, and RETDATA to carry the data. In bus-based interfaces RETDATA and RETSTAT typically exist as two separate signals [1, 12, 13, 17, 18, 26]. Messages composing a transaction are divided in outgoing messages, namely CMD and OUTDATA, and response messages, namely RETDATA, RETSTAT. Within a transaction, CMD precedes all other messages, and RETDATA precedes RETSTAT if present. These rules apply both between master and ANIP, and PNIP and slave. Examples of transactions are shown in Figure 5. We classify connections as follows (see Figure 6): • A simple connection is a connection between one ANIP and one PNIP. • A narrowcast connection is a connection between one ANIP and one or more PNIPs, in which each transaction that the ANIP initiates is executed by exactly one PNIP. An example of the narrowcast connection is shown in Figure 7, where the ANIP performs transactions on an address space which is mapped on two memory modules. Depending on the transaction address, a transaction is executed on only one of these two memories.

'800 Patent Claim Limitation	Radulescu I
	A multicast connection is a connection between one ANIP and one or more PNIPs, in which the sent messages are duplicated and each PNIP receives a copy of those messages. In a multicast connection no return messages are currently allowed, because of the large traffic they generate (i.e., one response per destination). It could also increase the complexity in the ANIP because individual responses from PNIPs must be merged into a single response for the ANIP. This requires buffer space and/or additional computation for the merging itself.”  <pre> sequenceDiagram participant ANIP participant PNIP ANIP->>PNIP: OUTDATA PNIP->>ANIP: RETDATA ANIP->>PNIP: CMD PNIP->>ANIP: RETSTAT </pre> <i>Figure 4.</i> Transaction composition.  <pre> sequenceDiagram participant ANIP participant PNIP ANIP->>PNIP: DATA ANIP->>PNIP: WRITE Note over ANIP,PNIP: unacknowledged write ANIP->>PNIP: READ PNIP->>ANIP: DATA Note over ANIP,PNIP: read ANIP->>PNIP: DATA ANIP->>PNIP: WRITE PNIP->>ANIP: OK / ERR Note over ANIP,PNIP: acknowledged write ANIP->>PNIP1: WRITE ANIP->>PNIP2: WRITE Note over ANIP,PNIP1,PNIP2: multicast </pre> <i>Figure 5.</i> Transaction examples.

'800 Patent Claim Limitation	Radulescu I
	simple: ANIP ↔ PNIP narrowcast: ANIP → PNIP ANIP → PNIP ANIP → PNIP multicast: ANIP → PNIP ANIP → PNIP ANIP → PNIP <i>Figure 7.</i> A narrowcast connection. <i>Figure 6.</i> Connection types. See, e.g., p. 292: “There are several points in a connection where order of transactions can be observed (see Figure 8): (a) the order in which the master presents CMD messages to the ANIP, (b) the order in which the CMDs are delivered to the slave by the PNIP, (c) the order in which the slave presents the responses to the PNIP, and (d) the order the responses are delivered to the master by the ANIP. Note that not all of (b), (c), and (d) are always present. Moreover, there are no assumptions about the order in which the slaves execute transactions; we can only observe the order of the responses. We consider the order of the transaction execution by the slaves to be a system decision, and not a part of the interconnect protocol.”

'800 Patent Claim Limitation	Radulescu I
	 <i>Figure 8.</i> Message ordering is observable at a, b, c, and d. To the extent Radulescu I does not expressly disclose this limitation, a person of ordinary skill in the art would have determined that this limitation is either inherent and/or obvious to one of ordinary skill in the art in view of the teachings of Radulescu I. Further, one of ordinary skill in the art would have been motivated to modify Radulescu I or combine it with any of the present prior art references found in Defendant’s Invalidity Contentions and any supplements thereto and the relevant section charts for other prior art for the ’800 Patent in a manner that would result in the subject matter of this limitation given, at the very least, the person’s understanding of the state of the art, the problems addressed and solved in the prior art, and the teachings of Radulescu I.
[12.c] a network for connecting the first interconnect module to the second interconnect module in order to provide a communication connection between the first processing module and the second processing module, wherein the first interconnect module is connected between the first processing module and the network, and wherein the	Radulescu I discloses and/or renders obvious the preamble based on at least the following citations. See, e.g.: <i>See, e.g.,</i> p. 280-81: “Introducing networks (Figure 1) as on-chip interconnects radically changes the communication as compared to direct interconnects such as buses or switches (Figure 2). This is because of the multi-hop nature of a network, where communication modules are not directly connected, but separated by one or more network nodes. This is in contrast with the prevalent existing interconnects (i.e., buses) where modules are directly connected. The implications of this change reside in the arbitration (which must change from centralized to distributed), and in the communication properties (e.g., ordering, or flow control). In this section, we list some of these topics, and outline the differences between NoCs and buses. We refer mainly to buses as direct interconnects, because currently they are the most used on-chip interconnect. Most of the bus characteristics also hold for other direct interconnects (e.g.,

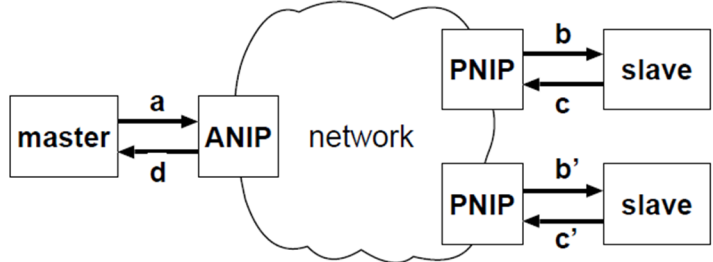
'800 Patent Claim Limitation	Radulescu I
second interconnect module is connected between the second processing module and the network;	switches [16]). Multilevel buses are a hybrid between buses and NoCs. For our purposes, depending on the functionality of the bridges, multilevel buses behave either like simple buses [2] or like NoCs.”  <i>Figure 1.</i> A network interconnect example  <i>Figure 2.</i> A bus interconnect example See, e.g., p. 287-88: “IPs interact with our network [8, 20, 21] at so-called network interfaces (NI). NIs provide NI ports (NIP) through which the communication services are accessed. As shown in Figure 3, a NI can have several NIPs to which one or more IPs (computation elements or memories, but not interconnection elements) can be connected. Similarly, an IP can be connected to more than one NI and NIP.”  <i>Figure 3.</i> Examples of links between NIs and IPs. See, e.g., p. 289-91: “A transaction is composed from the following messages (see Figure 4):

'800 Patent Claim Limitation	Radulescu I
	• A command message (CMD) is sent by the ANIP, and describes the action to be executed at the slave connected to the PNIP. Examples of commands are read, write, test and set, and flush. Commands are the only messages that are compulsory in a transaction. For NIPs that allow only a single command with no parameters (e.g., fixed-size address-less write), we assume the command message still exists, even if it is implicit (i.e., not explicitly sent by the IP). • An out data message (OUTDATA) is sent by the ANIP following a command that requires data to be executed (e.g., write, multicast, and test-and-set). • A return data message (RETDATA) is sent by a PNIP as a consequence of a transaction execution that produces data (e.g., read, and test-and-set). • A completion acknowledgment message (RETSTAT) is an optional message which is returned by PNIP when a command has been completed. It may signal either a successful completion or an error. For transactions including both RETDATA and RETSTAT the two messages can be combined in a single message for efficiency. However, conceptually, they exist both: RETSTAT to signal the presence of data or an error, and RETDATA to carry the data. In bus-based interfaces RETDATA and RETSTAT typically exist as two separate signals [1, 12, 13, 17, 18, 26]. Communication takes place on connections using transactions, consisting of a request and possibly a response. The request encodes an operation (e.g., read, write, flush, test and set, nop) and possibly carries outgoing data (e.g., for write commands). The response returns data as a result of a command (e.g., read) and/or an acknowledgment. Connections involve at least two NIPs. Transactions on a connection are always started at one and only one of the NIPs, called the connection's active NIP (ANIP). All the other NIPs of the connection are called passive NIPs (PNIP). There can be multiple transactions active on a connection at a time, but more generally than for split buses. That is, transactions can be started at the ANIP of a connection while responses for earlier transactions are pending. If a connection has multiple slaves, multiple transactions can be initiated towards different slaves. Transactions are also pipelined between a single master-slave pair for both

'800 Patent Claim Limitation	Radulescu I
	requests and responses. In principle, transactions can also be pipelined within a slave, if the slave allows this. A transaction is composed from the following messages (see Figure 4): • A command message (CMD) is sent by the ANIP, and describes the action to be executed at the slave connected to the PNIP. Examples of commands are read, write, test and set, and flush. Commands are the only messages that are compulsory in a transaction. For NIPs that allow only a single command with no parameters (e.g., fixed-size address-less write), we assume the command message still exists, even if it is implicit (i.e., not explicitly sent by the IP). • An out data message (OUTDATA) is sent by the ANIP following a command that requires data to be executed (e.g., write, multicast, and test-and-set). • A return data message (RETDATA) is sent by a PNIP as a consequence of a transaction execution that produces data (e.g., read, and test-and-set). • A completion acknowledgment message (RETSTAT) is an optional message which is returned by PNIP when a command has been completed. It may signal either a successful completion or an error. For transactions including both RETDATA and RETSTAT the two messages can be combined in a single message for efficiency. However, conceptually, they exist both: RETSTAT to signal the presence of data or an error, and RETDATA to carry the data. In bus-based interfaces RETDATA and RETSTAT typically exist as two separate signals [1, 12, 13, 17, 18, 26]. Messages composing a transaction are divided in outgoing messages, namely CMD and OUTDATA, and response messages, namely RETDATA, RETSTAT. Within a transaction, CMD precedes all other messages, and RETDATA precedes RETSTAT if present. These rules apply both between master and ANIP, and PNIP and slave. Examples of transactions are shown in Figure 5. We classify connections as follows (see Figure 6): • A simple connection is a connection between one ANIP and one PNIP. • A narrowcast connection is a connection between one ANIP and one or more PNIPs, in which each transaction that the ANIP initiates is executed by exactly one PNIP. An example of the narrowcast connection is shown in Figure 7, where the ANIP performs transactions on

'800 Patent Claim Limitation	Radulescu I
	an address space which is mapped on two memory modules. Depending on the transaction address, a transaction is executed on only one of these two memories. • A multicast connection is a connection between one ANIP and one or more PNIPs, in which the sent messages are duplicated and each PNIP receives a copy of those messages. In a multicast connection no return messages are currently allowed, because of the large traffic they generate (i.e., one response per destination). It could also increase the complexity in the ANIP because individual responses from PNIPs must be merged into a single response for the ANIP. This requires buffer space and/or additional computation for the merging itself.” <pre> sequenceDiagram participant ANIP participant PNIP ANIP->>PNIP: OUTDATA PNIP->>ANIP: RETDATA ANIP->>PNIP: CMD PNIP->>ANIP: RETSTAT </pre> <i>Figure 4.</i> Transaction composition. <pre> sequenceDiagram participant ANIP participant PNIP ANIP->>PNIP: DATA PNIP->>ANIP: WRITE Note over ANIP,PNIP: unacknowledged write ANIP->>PNIP: READ PNIP->>ANIP: DATA Note over ANIP,PNIP: read ANIP->>PNIP: DATA PNIP->>ANIP: WRITE PNIP->>ANIP: OK / ERR Note over ANIP,PNIP: acknowledged write ANIP->>PNIP1: WRITE ANIP->>PNIP2: WRITE Note over ANIP,PNIP1,PNIP2: multicast </pre> <i>Figure 5.</i> Transaction examples.

'800 Patent Claim Limitation	Radulescu I
	simple: ANIP ↔ PNIP narrowcast: ANIP → PNIP ANIP → PNIP ANIP → PNIP multicast: ANIP → PNIP ANIP → PNIP ANIP → PNIP <i>Figure 7.</i> A narrowcast connection. <i>Figure 6.</i> Connection types. <i>See, e.g., p. 292:</i> “There are several points in a connection where order of transactions can be observed (see Figure 8): (a) the order in which the master presents CMD messages to the ANIP, (b) the order in which the CMDs are delivered to the slave by the PNIP, (c) the order in which the slave presents the responses to the PNIP, and (d) the order the responses are delivered to the master by the ANIP. Note that not all of (b), (c), and (d) are always present. Moreover, there are no assumptions about the order in which the slaves execute transactions; we can only observe the order of the responses. We consider the order of the transaction execution by the slaves to be a system decision, and not a part of the interconnect protocol.”

'800 Patent Claim Limitation	Radulescu I
	 <i>Figure 8.</i> Message ordering is observable at a, b, c, and d. To the extent Radulescu I does not expressly disclose this limitation, a person of ordinary skill in the art would have determined that this limitation is either inherent and/or obvious to one of ordinary skill in the art in view of the teachings of Radulescu I. Further, one of ordinary skill in the art would have been motivated to modify Radulescu I or combine it with any of the present prior art references found in Defendant’s Invalidity Contentions and any supplements thereto and the relevant section charts for other prior art for the ’800 Patent in a manner that would result in the subject matter of this limitation given, at the very least, the person’s understanding of the state of the art, the problems addressed and solved in the prior art, and the teachings of Radulescu I.
[12.d] wherein the first interconnect module includes a first memory for buffering first data from the first processing module, the first interconnect module further including a first determination unit, wherein the first determination unit is configured to determine a first optimal amount of data to be buffered by the first memory, and the second	Radulescu I discloses and/or renders obvious the preamble based on at least the following citations. See, e.g.: <i>See, e.g.</i>, p. 285: “Buffering and Flow Control. Buffering data of a master (output buffering) is used both for buses and NoCs to decouple computation from communication. However, for NoCs output buffering is also needed to marshal data, which consists of (a) (optionally) splitting the outgoing data in smaller packets which are transported by the network, and (b) adding control information for the network around the data (packet header). To avoid output buffer overflow the master must not initiate transactions that generate more data than the currently available space. Similarly to output buffering, input buffering is also used to decouple computation from communication. In a NoC, input buffering is also required to unmarshal data. In addition, flow control for input buffers differs for buses and NoCs. For buses, the source and destination are directly linked, and, destination can therefore signal directly to a source that it

'800 Patent Claim Limitation	Radulescu I
interconnect module includes a second memory for buffering second data from the second processing module, the second interconnect module further including a second determination unit, wherein the second determination unit is configured to determine a second optimal amount of data to be buffered by the second memory,	cannot accept data. This information can even be available to the arbiter such that the bus is not granted to a transaction trying to write to a full buffer. In a NoC, however, the destination of a transaction cannot signal directly to a source that its input buffer is full. Consequently, transactions to a destination can be started, possibly from multiple sources, after the destination's input buffer has filled up. Several policies can be adopted when an input buffer is full. One policy is not to accept additional incoming transitions, and to store them in the network. However, this approach can easily lead to network congestion, as the data could be eventually stored all the way to the sources, blocking the links in between. Another policy is to accept incoming transactions at a full destination, and drop some data in the input buffer. Congestion is avoided but data is lost, and this is undesirable. To avoid input buffer overflow connections can be used, together with end-to-end flow control. At connection set up between a master and one or more slaves, buffer space is allocated at the network interfaces of the slaves, and the network interface of the master is assigned credits reflecting the amount of buffer space at the slaves. The master can only send data when it has enough credits for the destination slave(s). The slaves grant credits to the master when they consume data." <i>See, e.g., p. 288:</i> "Æthereal connections must be created with the desired properties before being used. This may result in resource reservations inside the network (e.g., buffer space, or percentage of the link usage per time unit). If the requested resources are not available, the network will refuse the request. After usage, connections are closed, which leads to freeing the resources occupied by that connection." <i>See, e.g., p. 288:</i> "To allow more flexibility in configuring connections and, hence, better resource allocation per connection, the outgoing and return parts of connections are configured independently. For example, a different amount of buffer space can be allocated in the NIPs at master and slaves, or different bandwidths can be reserved for requests and responses." <i>See, e.g., p. 289:</i> "Connections involve at least two NIPs. Transactions on a connection are always started at one and only one of the NIPs, called the connection's active NIP (ANIP). All the other NIPs of the connection are called passive NIPs (PNIP)."

'800 Patent Claim Limitation	Radulescu I
	<i>See, e.g.</i>, p. 295: “Connection latency, throughput, and jitter. In our network, throughput can be reserved for connections in a time-division multiple access (TDMA) fashion, where bandwidth is split in fixed-size slots on a fixed time frame. Bandwidth, as well as bounds on latency and jitter, can be guaranteed when slots are reserved. They are all defined in multiples of the slots. throughput, latency and jitter can all be configured independently for the request and response parts of a connection. Fully guaranteed-throughput connections (i.e., providing throughput guarantees on both request and return parts of the connection) can overbook resources in some cases. For example, when an ANIP opens a guaranteed-throughput read connection, it must reserve slots for the read command messages, and for the read data messages. The ratio between the two can be very large (e.g., 1:100), which leads either to a large number of slots, or bandwidth being wasted for the read command messages. To resolve this problem, the request part of a connection can be best effort, while the response can have guaranteed throughput (or vice versa). For the example mentioned above, one can use best effort read messages, and guaranteed-throughput read-data messages. No global connection guarantees can be offered in this case, but the overall throughput can be higher and more stable than in the case of using only best-effort traffic.” <i>See, e.g.</i>, p. 295-96: “Connection flow control. As mentioned earlier, our network guarantees that messages are delivered to the NI. Messages sent from one of the NIPs are not immediately visible at the other NIP, because of the multi-hop nature of networks. Consequently, handshakes over a network would allow only a single message be transmitted at a time. This limits the throughput on a connection and adds latency to transactions. To overcome this problem, and achieve a better network utilization, the messages must be pipelined. In this case, if the data is not consumed at the PNIP at the same rate it arrives, either flow control must be introduced to slow down the producer, or data may be lost because of limited buffer space at the consumer NI. We introduce end-to-end flow control at the level of connections, which requires buffer space to be associated with connections. End-to-end flow control ensures that messages are sent over the network only when there is enough space in the NIP’s destination buffer to accommodate them.

'800 Patent Claim Limitation	Radulescu I
	End-to-end flow is optional (i.e., to be requested when connections are opened) and can be configured independently for the outgoing and return paths. When no flow control is provided, messages are dropped when buffers overflow. Multiple policies of dropping messages are possible, as in off-chip networks. Possible scenarios include: (a) the oldest message is dropped (milk policy), or (b) the newest message is dropped (wine policy) [25]. We opt for a credit-based flow control. Credits are associated with the empty buffer space at the receiver NI. The sender's credit is lowered as data is sent. When the PNIP delivers data to the slave, credits are granted to the sender. If the sender's credit is not sufficient to send some data, the NI at the sender stalls the sending." To the extent Radulescu I does not expressly disclose this limitation, a person of ordinary skill in the art would have determined that this limitation is either inherent and/or obvious to one of ordinary skill in the art in view of the teachings of Radulescu I. Further, one of ordinary skill in the art would have been motivated to modify Radulescu I or combine it with any of the present prior art references found in Defendant's Invalidity Contentions and any supplements thereto and the relevant section charts for other prior art for the '800 Patent in a manner that would result in the subject matter of this limitation given, at the very least, the person's understanding of the state of the art, the problems addressed and solved in the prior art, and the teachings of Radulescu I.
[12.e] the first interconnect module transferring the first data to the second processing module when the first data buffered in the first memory reaches the first optimal amount, and the second interconnect module transferring the second data to the first processing module when the second data buffered in the second	Radulescu I discloses and/or renders obvious the preamble based on at least the following citations. See, e.g.: <i>See, e.g.</i>, p. 285: "Buffering and Flow Control. Buffering data of a master (output buffering) is used both for buses and NoCs to decouple computation from communication. However, for NoCs output buffering is also needed to marshal data, which consists of (a) (optionally) splitting the outgoing data in smaller packets which are transported by the network, and (b) adding control information for the network around the data (packet header). To avoid output buffer overflow the master must not initiate transactions that generate more data than the currently available space. Similarly to output buffering, input buffering is also used to decouple computation from communication. In a NoC, input buffering is also required to unmarshal data. In addition, flow control for input buffers differs for buses and NoCs. For buses, the source and destination are directly linked, and, destination can therefore signal directly to a source that it

'800 Patent Claim Limitation	Radulescu I
memory reaches the second optimal amount	cannot accept data. This information can even be available to the arbiter such that the bus is not granted to a transaction trying to write to a full buffer. In a NoC, however, the destination of a transaction cannot signal directly to a source that its input buffer is full. Consequently, transactions to a destination can be started, possibly from multiple sources, after the destination's input buffer has filled up. Several policies can be adopted when an input buffer is full. One policy is not to accept additional incoming transitions, and to store them in the network. However, this approach can easily lead to network congestion, as the data could be eventually stored all the way to the sources, blocking the links in between. Another policy is to accept incoming transactions at a full destination, and drop some data in the input buffer. Congestion is avoided but data is lost, and this is undesirable. To avoid input buffer overflow connections can be used, together with end-to-end flow control. At connection set up between a master and one or more slaves, buffer space is allocated at the network interfaces of the slaves, and the network interface of the master is assigned credits reflecting the amount of buffer space at the slaves. The master can only send data when it has enough credits for the destination slave(s). The slaves grant credits to the master when they consume data." <i>See, e.g., p. 288:</i> "To allow more flexibility in configuring connections and, hence, better resource allocation per connection, the outgoing and return parts of connections are configured independently. For example, a different amount of buffer space can be allocated in the NIPs at master and slaves, or different bandwidths can be reserved for requests and responses." <i>See, e.g., p. 289:</i> "Connections involve at least two NIPs. Transactions on a connection are always started at one and only one of the NIPs, called the connection's active NIP (ANIP). All the other NIPs of the connection are called passive NIPs (PNIP)." <i>See, e.g., p. 295-96:</i> "Connection flow control. As mentioned earlier, our network guarantees that messages are delivered to the NI. Messages sent from one of the NIPs are not immediately visible at the other NIP, because of the multi-hop nature of networks. Consequently, handshakes over a network would allow only a single message be transmitted at a time. This limits the throughput on a connection and adds latency to transactions. To overcome this problem, and achieve a better

'800 Patent Claim Limitation	Radulescu I
	network utilization, the messages must be pipelined. In this case, if the data is not consumed at the PNIP at the same rate it arrives, either flow control must be introduced to slow down the producer, or data may be lost because of limited buffer space at the consumer NI. We introduce end-to-end flow control at the level of connections, which requires buffer space to be associated with connections. End-to-end flow control ensures that messages are sent over the network only when there is enough space in the NIP's destination buffer to accommodate them. End-to-end flow is optional (i.e., to be requested when connections are opened) and can be configured independently for the outgoing and return paths. When no flow control is provided, messages are dropped when buffers overflow. Multiple policies of dropping messages are possible, as in off-chip networks. Possible scenarios include: (a) the oldest message is dropped (milk policy), or (b) the newest message is dropped (wine policy) [25]. We opt for a credit-based flow control. Credits are associated with the empty buffer space at the receiver NI. The sender's credit is lowered as data is sent. When the PNIP delivers data to the slave, credits are granted to the sender. If the sender's credit is not sufficient to send some data, the NI at the sender stalls the sending." To the extent Radulescu I does not expressly disclose this limitation, a person of ordinary skill in the art would have determined that this limitation is either inherent and/or obvious to one of ordinary skill in the art in view of the teachings of Radulescu I. Further, one of ordinary skill in the art would have been motivated to modify Radulescu I or combine it with any of the present prior art references found in Defendant's Invalidity Contentions and any supplements thereto and the relevant section charts for other prior art for the '800 Patent in a manner that would result in the subject matter of this limitation given, at the very least, the person's understanding of the state of the art, the problems addressed and solved in the prior art, and the teachings of Radulescu I.
[12.f] wherein at least one of the first determination unit and the second determination unit is further configured to determine an optimal moment for sending the data	Radulescu I discloses and/or renders obvious the preamble based on at least the following citations. See, e.g.: <i>See, e.g.</i>, p. 275: "Networks are emerging as a possible solution for future on-chip interconnects. In this chapter, we show how networks on chip (NoC) are similar to and differ from both off-chip networks (e.g., computer networks) and current on-chip interconnects (e.g., buses). We re-examine

'800 Patent Claim Limitation	Radulescu I
in said first wrapper or said second wrapper according to communication properties of the communication between the master and the slave,	their communication services in the context of NoCs. To enable a clean separation between the NoC and IP blocks, we provide services that abstract from network implementations. We define a request-response transaction model similar to bus protocols, to make our approach backward compatible. To exploit the full power of NoCs, we also provide connection-oriented communication with differentiated services. Examples are bandwidth guarantees, transaction orderings, and end-to-end flow control.” <i>See, e.g., p. 276-77:</i> “In this chapter, we compare NoCs and off-chip networks showing both their similarities and differences. We also explore the differences between NoCs and existing on-chip interconnects. We present an interface which takes these issues into consideration. Our interface is aimed at being similar to a split-transaction bus interface, such as VCI [26], OCP [17] or DTL [18], to allow simple, low-cost wrappers to bus interfaces, and to allow backward compatibility with existing IPs. Our interface uses a request-response protocol that provides basic read and write operations. But it extends bus interfaces to fully exploit the power of our NoC [8, 20, 21]. For example, it offers connection-based communication where end-to-end flow control and time-related guarantees (e.g., bounded latency) can be requested.” <i>See, e.g., p. 278:</i> “In the rest of the section, we list five network issues that have a direct impact on the NoC cost: reliable communication, deadlock, data ordering, network flow control and buffering strategy, and time-related guarantees. For each of them, we discuss the differences and similarities for on- and off-chip networks.” <i>See, e.g., p. 278:</i> “Reliable communication. A consequence of the tight on-chip resource constraints is that the network components (i.e., routers and network interfaces) must be fairly simple to minimize computation and memory requirements. Luckily, on-chip wires currently provide a reliable communication medium, which can help to avoid the considerable overhead incurred by off-chip networks for providing reliable communication. Data integrity can be provided at low cost at the data link layer. However, data loss also depends on the network architecture. In most computer networks data is simply dropped if congestion occurs in the network [6,25]. On-chip, dropping data may lead to a too costly implementation of reliable communication. We show below that a network

'800 Patent Claim Limitation	Radulescu I
	that does not drop data can be a much lower-cost solution, at the peril of introducing the possibility of deadlock.” <i>See, e.g.</i>, p. 278-79: “Deadlock. Computer network topologies have generally an irregular (possibly dynamic) structure, which can introduce buffer cycles. In such topologies, packet dropping at the network nodes may be required to avoid deadlocks. Deadlock can also be avoided without dropping data, for example by introducing constraints either in the topology or routing. Fat-tree topologies have already been considered for NoCs, where deadlock is avoided by bouncing back packets in the network in case of buffer overflow [9]. Tile-based approaches to system design [7, 15, 24] use mesh or torus network topologies, where deadlock can be avoided using, for example, a turn-model routing algorithm [6]. An alternative solution for deadlock in NoCs, which takes into consideration that modules connecting to the network are either masters (initiating requests and receiving responses), or slaves (receiving requests and sending back responses), is to maintain separate virtual networks (with separate buffers) for requests and responses [6].” <i>See, e.g.</i>, p. 279: “Data ordering. In a network, data sent from a source to a destination may arrive out of order due to reordering in network nodes, following different routes, or retransmission after dropping. For off-chip networks out-of-order data delivery is typical. However, for NoCs where no data is dropped, data can be forced to follow the same path between a source and a destination (deterministic routing) with no reordering. This in-order data transportation requires less buffer space, and reordering modules are no longer necessary.” <i>See, e.g.</i>, p. 279-80: “Network flow control and buffering strategy. Network flow control and buffering strategy have a direct impact on the memory utilization in the network. Wormhole routing requires only a flit buffer (per queue) in the router, whereas store-and-forward and virtual-cut-through routing require at least the buffer space to accommodate a packet [6]. Consequently, on chip, wormhole routing may be preferred over virtual-cut-through or store-and-forward routing. Similarly, input queuing may be a lower memory-cost alternative to virtual-output-queuing or output-queuing buffering strategies, because it has fewer queues. Dedicated (lower cost) fifo memory structures also enable on-chip usage of virtual-cut-through routing or virtual output queuing

'800 Patent Claim Limitation	Radulescu I
	for a better performance [20]. However, using virtual-cut-through routing and virtual output queuing at the same time is still too costly [20].” <i>See, e.g.</i>, p. 280: “Time-related guarantees. Off-chip networks typically use packet switching and offer best-effort services. Contention can occur at each network node, making latency guarantees very hard to offer. Throughput guarantees can still be offered using schemes such as rate-based switching [27] or deadline-based packet switching [19], but with high buffering costs. An alternative to provide such time-related guarantees is to use time-division multiple access (TDMA) circuits, where every circuit is dedicated to a network connection. Circuits provide guarantees at a relatively low memory and computation cost. Network resource utilization is increased when the network architecture allows any left-over guaranteed bandwidth to be used by best-effort communication [10, 20, 21].” <i>See, e.g.</i>, p. 282: “Transaction Ordering. Traditionally, all transactions on a bus are ordered (cf. Peripheral VCI [26], AMBA [1], DTL [18], or CoreConnect PLB and OPB [12, 13]). This is possible at a low cost, because the interconnect, being a direct link between the communicating parties, does not reorder data. However, on a split bus, a total ordering of transactions on a single master may still cause performance penalties, when slaves respond at different speeds. To solve this problem, recent extensions to bus protocols allow transactions to be performed on connections. Ordering of transactions within a connection is still preserved, but between connections there are no ordering constraints (e.g., OCP [17], or Basic VCI [26]). A few of the bus protocols allow out-of-order responses per connection in their advanced modes (e.g., Advanced VCI [26]), but both requests and responses arrive at the destination in the same order as they were sent. In a NoC, ordering becomes weaker. Global ordering can only be provided at a very high cost due to the conflict between the distributed nature of the networks, and the requirement of a centralized arbitration necessary for global ordering. Even local ordering, between a source-destination pair, may be costly. Data may arrive out of order if it is transported over multiple routes. In such cases, to still achieve an in-order delivery, data must be labeled with sequence numbers and reordered at the destination before being delivered.”

'800 Patent Claim Limitation	Radulescu I
	<i>See, e.g.</i>, p. 284: “Latency. Transaction latency is caused by two factors: (a) the access time to the bus, which is the time until the bus is granted, and (b) the latency introduced by the interconnect to transfer the data. For a bus, where the arbitration is centralized, the access time is proportional to the number of masters connected to the bus. The transfer latency itself typically is constant and relatively low, because the modules are linked directly. However, the speed of transfer is limited by the bus speed, which is relatively low. In a NoC, arbitration is performed at each router for the following link. The access time per router is small. Both end-to-end access time and transport time increase proportionally to the number of hops between master and slave. However, network links are unidirectional and point to point and, hence, can run at higher frequencies than buses, thus lowering the latency. From a latency prospective, using a bus or a network is a trade off between the number of modules connected to the interconnect (which affects access time), the speed of the interconnect, and the network topology.” <i>See, e.g.</i>, p. 285: “Buffering and Flow Control. Buffering data of a master (output buffering) is used both for buses and NoCs to decouple computation from communication. However, for NoCs output buffering is also needed to marshal data, which consists of (a) (optionally) splitting the outgoing data in smaller packets which are transported by the network, and (b) adding control information for the network around the data (packet header). To avoid output buffer overflow the master must not initiate transactions that generate more data than the currently available space. Similarly to output buffering, input buffering is also used to decouple computation from communication. In a NoC, input buffering is also required to unmarshal data. In addition, flow control for input buffers differs for buses and NoCs. For buses, the source and destination are directly linked, and, destination can therefore signal directly to a source that it cannot accept data. This information can even be available to the arbiter such that the bus is not granted to a transaction trying to write to a full buffer. In a NoC, however, the destination of a transaction cannot signal directly to a source that its input buffer is full. Consequently, transactions to a destination can be started, possibly from multiple sources, after the destination’s input buffer has filled up. Several policies can be adopted when an input buffer is full. One policy is not to accept additional incoming transitions, and to store them in the network. However, this approach can easily lead to network congestion, as the data could be

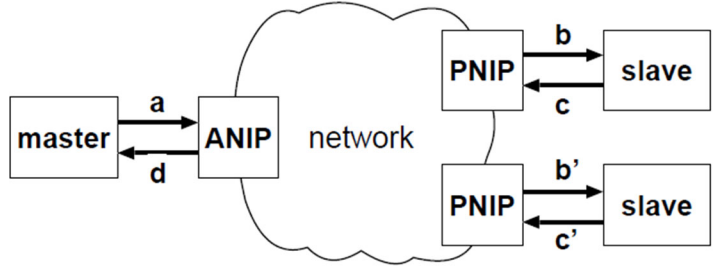
'800 Patent Claim Limitation	Radulescu I
	eventually stored all the way to the sources, blocking the links in between. Another policy is to accept incoming transactions at a full destination, and drop some data in the input buffer. Congestion is avoided but data is lost, and this is undesirable. To avoid input buffer overflow connections can be used, together with end-to-end flow control. At connection set up between a master and one or more slaves, buffer space is allocated at the network interfaces of the slaves, and the network interface of the master is assigned credits reflecting the amount of buffer space at the slaves. The master can only send data when it has enough credits for the destination slave(s). The slaves grant credits to the master when they consume data.” <i>See, e.g.</i>, p. 287: “Third, our interface extends traditional bus interfaces to fully exploit the power of NoCs. For example, we offer connection-based communication which does not only relax ordering constraints (as for buses), but also enables new communication properties, such as end-to-end flow control based on credits, or guaranteed throughput [8, 20, 21]. All these properties can be set for each connection individually.” <i>See, e.g.</i>, p. 288: “Communication between NIPs is performed on connections. Connections are introduced to describe and identify communication with different properties, such as guaranteed throughput, bounded latency and jitter, ordered delivery, or flow control. For example, to distinguish and independently guarantee communication of 1Mbs and 25Mbs, two connections can be used. Two NIPs can be connected by multiple connections, possibly with different properties. Connections as defined here are similar to the concept of threads and connections from OCP and VCI. Where in OCP and VCI connections are used only to relax transaction ordering, we generalize from only the ordering property to include configuration of buffering and flow control, guaranteed throughput, and bounded latency per connection.” <i>See, e.g.</i>, p. 288: “To allow more flexibility in configuring connections and, hence, better resource allocation per connection, the outgoing and return parts of connections are configured independently. For example, a different amount of buffer space can be allocated in the NIPs at master and slaves, or different bandwidths can be reserved for requests and responses.”

'800 Patent Claim Limitation	Radulescu I
	<i>See, e.g.</i>, p. 288: “Depending on the requested services, the time to handle a connection (i.e., creating, closing, modifying services) can be short (e.g., creating/closing an unordered, lossy, best-effort connection) or significant (e.g., creating/closing a multicast guaranteed-throughput connection). Consequently, connections are assumed to be created, closed, or modified infrequently, coinciding e.g., with reconfiguration points, when the application requirements change.” <i>See, e.g.</i>, p. 289: “Connections involve at least two NIPs. Transactions on a connection are always started at one and only one of the NIPs, called the connection’s active NIP (ANIP). All the other NIPs of the connection are called passive NIPs (PNIP).” <i>See, e.g.</i>, p. 291-96: “B. Connection Properties In this section we elaborate on the features that can be configured for a connection: guaranteed message integrity, guaranteed transaction completion, various transaction orderings, guaranteed throughput, bounded latency and jitter, and connection flow control. Data Integrity. Data integrity means that the content of messages is not changed (accidentally or not) during transport. We assume that data integrity is already solved at a lower layer in our network, namely at the link layer, because in current on-chip technologies data can be transported uncorrupted over links. Consequently, our network interface always guarantees that messages are delivered uncorrupted at the destination. Transaction Completion. A transaction without a response (e.g., a posted write) is said to be complete when it has been executed by the slave. As there is no response message to the master, no guarantee regarding transaction completion can be given. A transaction with a response (e.g., an acknowledged write) is said to be complete when a RETSTAT message is received from the ANIP1. The transaction may either (a) be executed successfully, in which case a success RETSTAT is returned, (b) fail in its execution at the slave, in which case an execution error RETSTAT is returned, or (c) fail because of buffer overflow in a connection with no flow control, in which case it reports an overflow error. We assume that when a slave accepts a CMD requesting a response, the slave always generates the response. In our network, routers do not drop data [21]. Therefore, messages are always guaranteed to be delivered at the NI. For connections with flow control, also NIs do not drop data. Thus, message delivery and, thus, transaction completion to the IPs is guaranteed automatically in this case.

'800 Patent Claim Limitation	Radulescu I
	However, if there is no flow control, messages may be dropped at the network interface in case of buffer overflow (see the paragraph on end-to-end flow control below). All of CMD, OUTDATA, and RETDATA may be dropped at the NI. To guarantee transaction completion, RETSTAT is not allowed to be dropped. Consequently, in the ANIPs enough buffer space must be provided to accommodate RETSTAT messages for all outstanding transactions. This is enforced by bounding the number of outstanding transactions. Transaction Ordering. Across different connections no ordering of transactions is defined at the transport layer. There are several points in a connection where order of transactions can be observed (see Figure 8): (a) the order in which the master presents CMD messages to the ANIP, (b) the order in which the CMDs are delivered to the slave by the PNIP, (c) the order in which the slave presents the responses to the PNIP, and (d) the order the responses are delivered to the master by the ANIP. Note that not all of (b), (c), and (d) are always present. Moreover, there are no assumptions about the order in which the slaves execute transactions; we can only observe the order of the responses. We consider the order of the transaction execution by the slaves to be a system decision, and not a part of the interconnect protocol. At both ANIP and PNIPs, outgoing messages belonging to different transactions on the same connection are allowed to be interleaved. For example, two write commands can be issued, and only afterwards their data. If the order of OUTDATA messages differs from the order of CMD messages, transaction identifiers must be introduced to associate OUTDATAs with their corresponding CMD. Outgoing messages can be delivered by the PNIPs to the slaves (see Figure 8-b) as follows: • Unordered, which imposes no order on the delivery of the outgoing messages of different transactions at the PNIPs. • Ordered locally, where transactions must be delivered to each PNIP in the order they were sent (Figure 8-a), but no order is imposed across PNIPs. Locally-ordered delivery of the outgoing messages can be provided either by an ordered data transportation, or by reordering outgoing messages at the PNIP. • Ordered globally, where transactions must be delivered in the order they were sent, across all PNIPs of the connection. Globally-ordered delivery of the outgoing part of transactions require a costly synchronization mechanism.

'800 Patent Claim Limitation	Radulescu I
	Transaction response messages can be delivered by the slaves to the PNIPs (see Figure 8-c) as follows: • Ordered, when RETDATA and RETSTAT messages are returned in the same order as the CMDs were delivered to the slave (Figure 8-b). • Unordered, otherwise. When responses are unordered, there has to be a mechanism to identify the transaction to which a response belongs. This is usually done using tags attached to messages for transaction identifications (similar to tags in VCI). Response messages can be delivered by the ANIP to the master (see Figure 8-d) as follows: • Unordered, which imposes no order on the delivery of responses. Here also, tags must be used to associate responses with their corresponding CMDs. • Ordered locally, where RETDATA and RETSTAT messages of transactions for a single slave are delivered in the order the original CMDs were presented by the master to the ANIP (Figure 8-a). Note that there is no ordering imposed for transactions to different slaves within the same connection. • Globally ordered, where all responses in a connection are delivered to the master in the same order as the original CMDs. When transactions are pipelined on a connection, then globally-ordered delivery of responses requires reordering at the ANIP. All $3 \times 2 \times 3 = 18$ combinations between the above orderings are possible. Out of these, we define and offer the following two. An unordered connection is a connection in which no ordering is assumed in any part of the transactions. As a result, the responses must be tagged to be able identify to which transaction they belong. Implementing unordered connections has low cost, however, they may be harder to use, and introduce the overhead of tagging. An ordered connection is defined as a connection with local ordering for the outgoing messages from PNIPs to slaves (Figure 8-b), ordered responses at the PNIPs (Figure 8-c), and global ordering for responses at the ANIP (Figure 8-d). We choose local ordering for the outgoing part because the global ordering has a too high cost, and has few uses. The ordering of responses is selected to allow a simple programming model with no tagging. Global ordering at the ANIP is possible at a moderate cost, because all the reordering is done locally in the ANIP.

'800 Patent Claim Limitation	Radulescu I
	A user can emulate connections with global ordering of outgoing and return messages at the PNIPs using non-pipelined acknowledged transactions, at the cost of high latency. Connection latency, throughput, and jitter. In our network, throughput can be reserved for connections in a time-division multiple access (TDMA) fashion, where bandwidth is split in fixed-size slots on a fixed time frame. Bandwidth, as well as bounds on latency and jitter, can be guaranteed when slots are reserved. They are all defined in multiples of the slots. throughput, latency and jitter can all be configured independently for the request and response parts of a connection. Fully guaranteed-throughput connections (i.e., providing throughput guarantees on both request and return parts of the connection) can overbook resources in some cases. For example, when an ANIP opens a guaranteed-throughput read connection, it must reserve slots for the read command messages, and for the read data messages. The ratio between the two can be very large (e.g., 1:100), which leads either to a large number of slots, or bandwidth being wasted for the read command messages. To resolve this problem, the request part of a connection can be best effort, while the response can have guaranteed throughput (or vice versa). For the example mentioned above, one can use best effort read messages, and guaranteed-throughput read-data messages. No global connection guarantees can be offered in this case, but the overall throughput can be higher and more stable than in the case of using only best-effort traffic. Connection flow control. As mentioned earlier, our network guarantees that messages are delivered to the NI. Messages sent from one of the NIPs are not immediately visible at the other NIP, because of the multi-hop nature of networks. Consequently, handshakes over a network would allow only a single message be transmitted at a time. This limits the throughput on a connection and adds latency to transactions. To overcome this problem, and achieve a better network utilization, the messages must be pipelined. In this case, if the data is not consumed at the PNIP at the same rate it arrives, either flow control must be introduced to slow down the producer, or data may be lost because of limited buffer space at the consumer NI. We introduce end-to-end flow control at the level of connections, which requires buffer space to be associated with connections. End-to-end flow control ensures that messages are sent over the network only when there is enough space in the NIP's destination buffer to accommodate them.

'800 Patent Claim Limitation	Radulescu I
	End-to-end flow is optional (i.e., to be requested when connections are opened) and can be configured independently for the outgoing and return paths. When no flow control is provided, messages are dropped when buffers overflow. Multiple policies of dropping messages are possible, as in off-chip networks. Possible scenarios include: (a) the oldest message is dropped (milk policy), or (b) the newest message is dropped (wine policy) [25]. We opt for a credit-based flow control. Credits are associated with the empty buffer space at the receiver NI. The sender's credit is lowered as data is sent. When the PNIP delivers data to the slave, credits are granted to the sender. If the sender's credit is not sufficient to send some data, the NI at the sender stalls the sending."  Figure 8. Message ordering is observable at a, b, c, and d. See, e.g., p. 296-97: "Video processing streams typically require a lossless, in-order video stream with guaranteed throughput, but possibly allow corrupted samples. An Æthereal connection for such a stream would require the necessary throughput, ordered transactions, and flow control. If the video stream is produced by the master, only write transactions are necessary. In this case, with a flow-controlled connection there is no need to also require transaction completion, because messages are never dropped, and the write command and its data are always delivered at the destination. Data integrity is always provided by our network, even though it may be not necessary in this case. Another example is that of cache updates which require uncorrupted, lossless, low-latency data transfer, but ordering and guaranteed throughput are less important. In this case, a connection would not require any time related guarantees, because even though an average low latency is required, a guarantee on low latency is not critical. Low latency can be obtained even with a best-

'800 Patent Claim Limitation	Radulescu I
	effort connection. The connection would also require flow control and guaranteed transaction completion to ensure lossless transactions. However, no ordering is necessary, because this is not important for cache updates, and allowing out of order transaction can reduce the response time.” <i>See, e.g.</i>, p. 297: “We also compare NoCs to existing on-chip interconnects, such as buses and switches. By directly connecting IP blocks, existing on-chip interconnects can offer tight coupling between masters and slaves, and global arbitration. In NoCs, masters and slaves are completely decoupled, and the arbitration is distributed over the network nodes. This make it harder to provide guarantees, such as bandwidth lower bounds, and transaction orderings.” <i>See, e.g.</i>, p. 297: “We define a set of NoC services that abstract from the network details. Using these services in IP design decouples computation and communication. We use a request-response transaction model to be close to existing on-chip interconnect protocols. This eases the migration of current IPs to NoCs. To fully utilize the NoC capabilities, such as high bandwidth and transaction concurrency, our services provide connection-oriented communication. Connections can be configured independently with different properties. These properties include transaction completion, various transaction orderings, bandwidth lower bounds, latency and jitter upper bounds, and flow control.” To the extent Radulescu I does not expressly disclose this limitation, a person of ordinary skill in the art would have determined that this limitation is either inherent and/or obvious to one of ordinary skill in the art in view of the teachings of Radulescu I. Further, one of ordinary skill in the art would have been motivated to modify Radulescu I or combine it with any of the present prior art references found in Defendant’s Invalidity Contentions and any supplements thereto and the relevant section charts for other prior art for the ’800 Patent in a manner that would result in the subject matter of this limitation given, at the very least, the person’s understanding of the state of the art, the problems addressed and solved in the prior art, and the teachings of Radulescu I.
[12.g] wherein the communication properties include ordering of data transport, flow control	Radulescu I discloses and/or renders obvious the preamble based on at least the following citations. <i>See, e.g.</i>:

'800 Patent Claim Limitation	Radulescu I
including when a remote buffer is reserved for a connection, then a data producer will be allowed to send data only when it is guaranteed that space is available for the produced data at the remote buffer, throughput where a lower bound on throughput is guaranteed, latency where an upper bound for latency is guaranteed, lossiness including dropping of data, transmission termination, transaction completion, data correctness, priority, and data delivery.	<i>See, e.g., p. 275:</i> “Networks are emerging as a possible solution for future on-chip interconnects. In this chapter, we show how networks on chip (NoC) are similar to and differ from both off-chip networks (e.g., computer networks) and current on-chip interconnects (e.g., buses). We re-examine their communication services in the context of NoCs. To enable a clean separation between the NoC and IP blocks, we provide services that abstract from network implementations. We define a request-response transaction model similar to bus protocols, to make our approach backward compatible. To exploit the full power of NoCs, we also provide connection-oriented communication with differentiated services. Examples are bandwidth guarantees, transaction orderings, and end-to-end flow control.” <i>See, e.g., p. 276-77:</i> “In this chapter, we compare NoCs and off-chip networks showing both their similarities and differences. We also explore the differences between NoCs and existing on-chip interconnects. We present an interface which takes these issues into consideration. Our interface is aimed at being similar to a split-transaction bus interface, such as VCI [26], OCP [17] or DTL [18], to allow simple, low-cost wrappers to bus interfaces, and to allow backward compatibility with existing IPs. Our interface uses a request-response protocol that provides basic read and write operations. But it extends bus interfaces to fully exploit the power of our NoC [8, 20, 21]. For example, it offers connection-based communication where end-to-end flow control and time-related guarantees (e.g., bounded latency) can be requested.” <i>See, e.g., p. 278:</i> “In the rest of the section, we list five network issues that have a direct impact on the NoC cost: reliable communication, deadlock, data ordering, network flow control and buffering strategy, and time-related guarantees. For each of them, we discuss the differences and similarities for on- and off-chip networks.” <i>See, e.g., p. 278:</i> “Reliable communication. A consequence of the tight on-chip resource constraints is that the network components (i.e., routers and network interfaces) must be fairly simple to minimize computation and memory requirements. Luckily, on-chip wires currently provide a reliable communication medium, which can help to avoid the considerable overhead incurred by off-chip networks for providing reliable communication. Data integrity can be provided at low cost at the data link layer. However, data loss also depends on the network architecture. In most computer

'800 Patent Claim Limitation	Radulescu I
	networks data is simply dropped if congestion occurs in the network [6,25]. On-chip, dropping data may lead to a too costly implementation of reliable communication. We show below that a network that does not drop data can be a much lower-cost solution, at the peril of introducing the possibility of deadlock.” <i>See, e.g.,</i> p. 278-79: “Deadlock. Computer network topologies have generally an irregular (possibly dynamic) structure, which can introduce buffer cycles. In such topologies, packet dropping at the network nodes may be required to avoid deadlocks. Deadlock can also be avoided without dropping data, for example by introducing constraints either in the topology or routing. Fat-tree topologies have already been considered for NoCs, where deadlock is avoided by bouncing back packets in the network in case of buffer overflow [9]. Tile-based approaches to system design [7, 15, 24] use mesh or torus network topologies, where deadlock can be avoided using, for example, a turn-model routing algorithm [6]. An alternative solution for deadlock in NoCs, which takes into consideration that modules connecting to the network are either masters (initiating requests and receiving responses), or slaves (receiving requests and sending back responses), is to maintain separate virtual networks (with separate buffers) for requests and responses [6].” <i>See, e.g.,</i> p. 279: “Data ordering. In a network, data sent from a source to a destination may arrive out of order due to reordering in network nodes, following different routes, or retransmission after dropping. For off-chip networks out-of-order data delivery is typical. However, for NoCs where no data is dropped, data can be forced to follow the same path between a source and a destination (deterministic routing) with no reordering. This in-order data transportation requires less buffer space, and reordering modules are no longer necessary.” <i>See, e.g.,</i> p. 279-80: “Network flow control and buffering strategy. Network flow control and buffering strategy have a direct impact on the memory utilization in the network. Wormhole routing requires only a flit buffer (per queue) in the router, whereas store-and-forward and virtual-cut-through routing require at least the buffer space to accommodate a packet [6]. Consequently, on chip, wormhole routing may be preferred over virtual-cut-through or store-and-forward routing. Similarly, input queuing may be a lower memory-cost alternative to virtual-output-queuing or

'800 Patent Claim Limitation	Radulescu I
	output-queuing buffering strategies, because it has fewer queues. Dedicated (lower cost) fifo memory structures also enable on-chip usage of virtual-cut-through routing or virtual output queuing for a better performance [20]. However, using virtual-cut-through routing and virtual output queuing at the same time is still too costly [20].” <i>See, e.g.</i>, p. 280: “Time-related guarantees. Off-chip networks typically use packet switching and offer best-effort services. Contention can occur at each network node, making latency guarantees very hard to offer. Throughput guarantees can still be offered using schemes such as rate-based switching [27] or deadline-based packet switching [19], but with high buffering costs. An alternative to provide such time-related guarantees is to use time-division multiple access (TDMA) circuits, where every circuit is dedicated to a network connection. Circuits provide guarantees at a relatively low memory and computation cost. Network resource utilization is increased when the network architecture allows any left-over guaranteed bandwidth to be used by best-effort communication [10, 20, 21].” <i>See, e.g.</i>, p. 282: “Transaction Ordering. Traditionally, all transactions on a bus are ordered (cf. Peripheral VCI [26], AMBA [1], DTL [18], or CoreConnect PLB and OPB [12, 13]). This is possible at a low cost, because the interconnect, being a direct link between the communicating parties, does not reorder data. However, on a split bus, a total ordering of transactions on a single master may still cause performance penalties, when slaves respond at different speeds. To solve this problem, recent extensions to bus protocols allow transactions to be performed on connections. Ordering of transactions within a connection is still preserved, but between connections there are no ordering constraints (e.g., OCP [17], or Basic VCI [26]). A few of the bus protocols allow out-of-order responses per connection in their advanced modes (e.g., Advanced VCI [26]), but both requests and responses arrive at the destination in the same order as they were sent. In a NoC, ordering becomes weaker. Global ordering can only be provided at a very high cost due to the conflict between the distributed nature of the networks, and the requirement of a centralized arbitration necessary for global ordering. Even local ordering, between a source-destination pair, may be costly. Data may arrive out of order if it is transported over multiple routes. In such cases, to still achieve an in-order delivery,

'800 Patent Claim Limitation	Radulescu I
	data must be labeled with sequence numbers and reordered at the destination before being delivered.” <i>See, e.g.</i>, p. 284: “Latency. Transaction latency is caused by two factors: (a) the access time to the bus, which is the time until the bus is granted, and (b) the latency introduced by the interconnect to transfer the data. For a bus, where the arbitration is centralized, the access time is proportional to the number of masters connected to the bus. The transfer latency itself typically is constant and relatively low, because the modules are linked directly. However, the speed of transfer is limited by the bus speed, which is relatively low. In a NoC, arbitration is performed at each router for the following link. The access time per router is small. Both end-to-end access time and transport time increase proportionally to the number of hops between master and slave. However, network links are unidirectional and point to point and, hence, can run at higher frequencies than buses, thus lowering the latency. From a latency prospective, using a bus or a network is a trade off between the number of modules connected to the interconnect (which affects access time), the speed of the interconnect, and the network topology.” <i>See, e.g.</i>, p. 285: “Buffering and Flow Control. Buffering data of a master (output buffering) is used both for buses and NoCs to decouple computation from communication. However, for NoCs output buffering is also needed to marshal data, which consists of (a) (optionally) splitting the outgoing data in smaller packets which are transported by the network, and (b) adding control information for the network around the data (packet header). To avoid output buffer overflow the master must not initiate transactions that generate more data than the currently available space. Similarly to output buffering, input buffering is also used to decouple computation from communication. In a NoC, input buffering is also required to unmarshal data. In addition, flow control for input buffers differs for buses and NoCs. For buses, the source and destination are directly linked, and, destination can therefore signal directly to a source that it cannot accept data. This information can even be available to the arbiter such that the bus is not granted to a transaction trying to write to a full buffer. In a NoC, however, the destination of a transaction cannot signal directly to a source that its input buffer is full. Consequently, transactions to a destination can be started, possibly from multiple

'800 Patent Claim Limitation	Radulescu I
	sources, after the destination's input buffer has filled up. Several policies can be adopted when an input buffer is full. One policy is not to accept additional incoming transitions, and to store them in the network. However, this approach can easily lead to network congestion, as the data could be eventually stored all the way to the sources, blocking the links in between. Another policy is to accept incoming transactions at a full destination, and drop some data in the input buffer. Congestion is avoided but data is lost, and this is undesirable. To avoid input buffer overflow connections can be used, together with end-to-end flow control. At connection set up between a master and one or more slaves, buffer space is allocated at the network interfaces of the slaves, and the network interface of the master is assigned credits reflecting the amount of buffer space at the slaves. The master can only send data when it has enough credits for the destination slave(s). The slaves grant credits to the master when they consume data." <i>See, e.g., p. 287:</i> "Third, our interface extends traditional bus interfaces to fully exploit the power of NoCs. For example, we offer connection-based communication which does not only relax ordering constraints (as for buses), but also enables new communication properties, such as end-to-end flow control based on credits, or guaranteed throughput [8, 20, 21]. All these properties can be set for each connection individually." <i>See, e.g., p. 288:</i> "Communication between NIPs is performed on connections. Connections are introduced to describe and identify communication with different properties, such as guaranteed throughput, bounded latency and jitter, ordered delivery, or flow control. For example, to distinguish and independently guarantee communication of 1Mbs and 25Mbs, two connections can be used. Two NIPs can be connected by multiple connections, possibly with different properties. Connections as defined here are similar to the concept of threads and connections from OCP and VCI. Where in OCP and VCI connections are used only to relax transaction ordering, we generalize from only the ordering property to include configuration of buffering and flow control, guaranteed throughput, and bounded latency per connection." <i>See, e.g., p. 288:</i> "To allow more flexibility in configuring connections and, hence, better resource allocation per connection, the outgoing and return parts of connections are configured

'800 Patent Claim Limitation	Radulescu I
	independently. For example, a different amount of buffer space can be allocated in the NIPs at master and slaves, or different bandwidths can be reserved for requests and responses.” <i>See, e.g.</i>, p. 288: “Depending on the requested services, the time to handle a connection (i.e., creating, closing, modifying services) can be short (e.g., creating/closing an unordered, lossy, best-effort connection) or significant (e.g., creating/closing a multicast guaranteed-throughput connection). Consequently, connections are assumed to be created, closed, or modified infrequently, coinciding e.g., with reconfiguration points, when the application requirements change.” <i>See, e.g.</i>, p. 289: “Connections involve at least two NIPs. Transactions on a connection are always started at one and only one of the NIPs, called the connection’s active NIP (ANIP). All the other NIPs of the connection are called passive NIPs (PNIP).” <i>See, e.g.</i>, p. 291-96: “B. Connection Properties In this section we elaborate on the features that can be configured for a connection: guaranteed message integrity, guaranteed transaction completion, various transaction orderings, guaranteed throughput, bounded latency and jitter, and connection flow control. Data Integrity. Data integrity means that the content of messages is not changed (accidentally or not) during transport. We assume that data integrity is already solved at a lower layer in our network, namely at the link layer, because in current on-chip technologies data can be transported uncorrupted over links. Consequently, our network interface always guarantees that messages are delivered uncorrupted at the destination. Transaction Completion. A transaction without a response (e.g., a posted write) is said to be complete when it has been executed by the slave. As there is no response message to the master, no guarantee regarding transaction completion can be given. A transaction with a response (e.g., an acknowledged write) is said to be complete when a RETSTAT message is received from the ANIP1. The transaction may either (a) be executed successfully, in which case a success RETSTAT is returned, (b) fail in its execution at the slave, in which case an execution error RETSTAT is returned, or (c) fail because of buffer overflow in a connection with no flow control, in which case it reports an overflow error. We assume that when a slave accepts a CMD requesting a response, the slave always generates the response.

'800 Patent Claim Limitation	Radulescu I
	In our network, routers do not drop data [21]. Therefore, messages are always guaranteed to be delivered at the NI. For connections with flow control, also NIs do not drop data. Thus, message delivery and, thus, transaction completion to the IPs is guaranteed automatically in this case. However, if there is no flow control, messages may be dropped at the network interface in case of buffer overflow (see the paragraph on end-to-end flow control below). All of CMD, OUTDATA, and RETDATA may be dropped at the NI. To guarantee transaction completion, RETSTAT is not allowed to be dropped. Consequently, in the ANIPs enough buffer space must be provided to accommodate RETSTAT messages for all outstanding transactions. This is enforced by bounding the number of outstanding transactions. Transaction Ordering. Across different connections no ordering of transactions is defined at the transport layer. There are several points in a connection where order of transactions can be observed (see Figure 8): (a) the order in which the master presents CMD messages to the ANIP, (b) the order in which the CMDs are delivered to the slave by the PNIP, (c) the order in which the slave presents the responses to the PNIP, and (d) the order the responses are delivered to the master by the ANIP. Note that not all of (b), (c), and (d) are always present. Moreover, there are no assumptions about the order in which the slaves execute transactions; we can only observe the order of the responses. We consider the order of the transaction execution by the slaves to be a system decision, and not a part of the interconnect protocol. At both ANIP and PNIPs, outgoing messages belonging to different transactions on the same connection are allowed to be interleaved. For example, two write commands can be issued, and only afterwards their data. If the order of OUTDATA messages differs from the order of CMD messages, transaction identifiers must be introduced to associate OUTDATAs with their corresponding CMD. Outgoing messages can be delivered by the PNIPs to the slaves (see Figure 8-b) as follows: • Unordered, which imposes no order on the delivery of the outgoing messages of different transactions at the PNIPs. • Ordered locally, where transactions must be delivered to each PNIP in the order they were sent (Figure 8-a), but no order is imposed across PNIPs. Locally-ordered delivery of the outgoing messages can be provided either by an ordered data transportation, or by reordering outgoing messages at the PNIP.

'800 Patent Claim Limitation	Radulescu I
	- Ordered globally, where transactions must be delivered in the order they were sent, across all PNIPs of the connection. Globally-ordered delivery of the outgoing part of transactions require a costly synchronization mechanism. Transaction response messages can be delivered by the slaves to the PNIPs (see Figure 8-c) as follows: - Ordered, when RETDATA and RETSTAT messages are returned in the same order as the CMDs were delivered to the slave (Figure 8-b). - Unordered, otherwise. When responses are unordered, there has to be a mechanism to identify the transaction to which a response belongs. This is usually done using tags attached to messages for transaction identifications (similar to tags in VCI). Response messages can be delivered by the ANIP to the master (see Figure 8-d) as follows: - Unordered, which imposes no order on the delivery of responses. Here also, tags must be used to associate responses with their corresponding CMDs. - Ordered locally, where RETDATA and RETSTAT messages of transactions for a single slave are delivered in the order the original CMDs were presented by the master to the ANIP (Figure 8-a). Note that there is no ordering imposed for transactions to different slaves within the same connection. - Globally ordered, where all responses in a connection are delivered to the master in the same order as the original CMDs. When transactions are pipelined on a connection, then globally-ordered delivery of responses requires reordering at the ANIP. All $3 \times 2 \times 3 = 18$ combinations between the above orderings are possible. Out of these, we define and offer the following two. An unordered connection is a connection in which no ordering is assumed in any part of the transactions. As a result, the responses must be tagged to be able identify to which transaction they belong. Implementing unordered connections has low cost, however, they may be harder to use, and introduce the overhead of tagging. An ordered connection is defined as a connection with local ordering for the outgoing messages from PNIPs to slaves (Figure 8-b), ordered responses at the PNIPs (Figure 8-c), and global ordering for responses at the ANIP (Figure 8-d). We choose local ordering for the outgoing part because the global ordering has a too high cost, and has few uses. The ordering of responses is

'800 Patent Claim Limitation	Radulescu I
	selected to allow a simple programming model with no tagging. Global ordering at the ANIP is possible at a moderate cost, because all the reordering is done locally in the ANIP. A user can emulate connections with global ordering of outgoing and return messages at the PNIPs using non-pipelined acknowledged transactions, at the cost of high latency. Connection latency, throughput, and jitter. In our network, throughput can be reserved for connections in a time-division multiple access (TDMA) fashion, where bandwidth is split in fixed-size slots on a fixed time frame. Bandwidth, as well as bounds on latency and jitter, can be guaranteed when slots are reserved. They are all defined in multiples of the slots. throughput, latency and jitter can all be configured independently for the request and response parts of a connection. Fully guaranteed-throughput connections (i.e., providing throughput guarantees on both request and return parts of the connection) can overbook resources in some cases. For example, when an ANIP opens a guaranteed-throughput read connection, it must reserve slots for the read command messages, and for the read data messages. The ratio between the two can be very large (e.g., 1:100), which leads either to a large number of slots, or bandwidth being wasted for the read command messages. To resolve this problem, the request part of a connection can be best effort, while the response can have guaranteed throughput (or vice versa). For the example mentioned above, one can use best effort read messages, and guaranteed-throughput read-data messages. No global connection guarantees can be offered in this case, but the overall throughput can be higher and more stable than in the case of using only best-effort traffic. Connection flow control. As mentioned earlier, our network guarantees that messages are delivered to the NI. Messages sent from one of the NIPs are not immediately visible at the other NIP, because of the multi-hop nature of networks. Consequently, handshakes over a network would allow only a single message be transmitted at a time. This limits the throughput on a connection and adds latency to transactions. To overcome this problem, and achieve a better network utilization, the messages must be pipelined. In this case, if the data is not consumed at the PNIP at the same rate it arrives, either flow control must be introduced to slow down the producer, or data may be lost because of limited buffer space at the consumer NI. We introduce end-to-end flow control at the level of connections, which requires buffer space to be associated with connections. End-to-end flow control ensures that messages are sent

'800 Patent Claim Limitation	Radulescu I
	over the network only when there is enough space in the NIP's destination buffer to accommodate them. End-to-end flow is optional (i.e., to be requested when connections are opened) and can be configured independently for the outgoing and return paths. When no flow control is provided, messages are dropped when buffers overflow. Multiple policies of dropping messages are possible, as in off-chip networks. Possible scenarios include: (a) the oldest message is dropped (milk policy), or (b) the newest message is dropped (wine policy) [25]. We opt for a credit-based flow control. Credits are associated with the empty buffer space at the receiver NI. The sender's credit is lowered as data is sent. When the PNIP delivers data to the slave, credits are granted to the sender. If the sender's credit is not sufficient to send some data, the NI at the sender stalls the sending." <pre> graph LR master[master] -- a --> ANIP[ANIP] ANIP -- d --> master ANIP --- network((network)) network --- PNIP1[PNIP] network --- PNIP2[PNIP] PNIP1 -- b --> slave1[slave] slave1 -- c --> PNIP1 PNIP2 -- b' --> slave2[slave] slave2 -- c' --> PNIP2 </pre> Figure 8. Message ordering is observable at a, b, c, and d. See, e.g., p. 296-97: "Video processing streams typically require a lossless, in-order video stream with guaranteed throughput, but possibly allow corrupted samples. An <i>Æthereal</i> connection for such a stream would require the necessary throughput, ordered transactions, and flow control. If the video stream is produced by the master, only write transactions are necessary. In this case, with a flow-controlled connection there is no need to also require transaction completion, because messages are never dropped, and the write command and its data are always delivered at the destination. Data integrity is always provided by our network, even though it may be not necessary in this case. Another example is that of cache updates which require uncorrupted, lossless, low-latency data transfer, but ordering and guaranteed throughput are less important. In this case, a connection

'800 Patent Claim Limitation	Radulescu I
	would not require any time related guarantees, because even though an average low latency is required, a guarantee on low latency is not critical. Low latency can be obtained even with a best-effort connection. The connection would also require flow control and guaranteed transaction completion to ensure lossless transactions. However, no ordering is necessary, because this is not important for cache updates, and allowing out of order transaction can reduce the response time.” <i>See, e.g.</i>, p. 297: “We also compare NoCs to existing on-chip interconnects, such as buses and switches. By directly connecting IP blocks, existing on-chip interconnects can offer tight coupling between masters and slaves, and global arbitration. In NoCs, masters and slaves are completely decoupled, and the arbitration is distributed over the network nodes. This make it harder to provide guarantees, such as bandwidth lower bounds, and transaction orderings.” <i>See, e.g.</i>, p. 297: “We define a set of NoC services that abstract from the network details. Using these services in IP design decouples computation and communication. We use a request-response transaction model to be close to existing on-chip interconnect protocols. This eases the migration of current IPs to NoCs. To fully utilize the NoC capabilities, such as high bandwidth and transaction concurrency, our services provide connection-oriented communication. Connections can be configured independently with different properties. These properties include transaction completion, various transaction orderings, bandwidth lower bounds, latency and jitter upper bounds, and flow control.” To the extent Radulescu I does not expressly disclose this limitation, a person of ordinary skill in the art would have determined that this limitation is either inherent and/or obvious to one of ordinary skill in the art in view of the teachings of Radulescu I. Further, one of ordinary skill in the art would have been motivated to modify Radulescu I or combine it with any of the present prior art references found in Defendant’s Invalidity Contentions and any supplements thereto and the relevant section charts for other prior art for the ’800 Patent in a manner that would result in the subject matter of this limitation given, at the very least, the person’s understanding of the state of the art, the problems addressed and solved in the prior art, and the teachings of Radulescu I.

