

**Exhibit E-5 to Defendants' Invalidity Contentions:
Comparison of U.S. Patent No. 8,072,893 ("2893 patent") and Xpipes: A Network-on-Chip Architecture for Gigascale Systems-on-Chips ("Xpipes")**

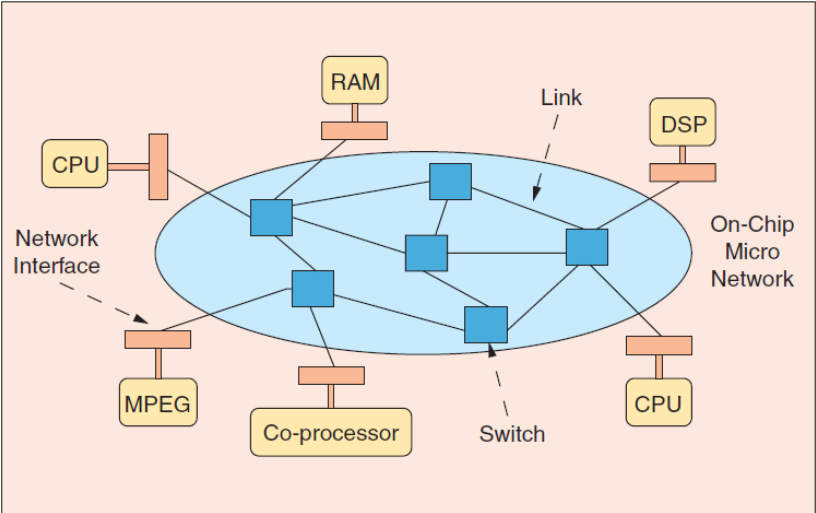
As described in the following claim chart, claim 10 of U.S. Patent No. 8,072,893 ("2893 patent") is invalid under 35 U.S.C. §§ 102 and/or 103 in view of Xpipes. The citations below are to Xpipes and compares Xpipes to the '2893 patent claims.¹

To the extent Plaintiff alleges that Xpipes does not disclose any particular limitation of claim 10 (the "Asserted Claim") of the '2893 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 '2893 patent to modify Xpipes and/or to combine the teachings of Xpipes 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 '2893 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 Xpipes would have been obvious for at least the following reasons: it would have been obvious to modify the methods and/or systems disclosed in Xpipes, alone or in combination with the references identified in Defendants' Invalidity Contentions, to meet this limitation based on: (a) the teachings in Xpipes itself; (b) a person of ordinary skill in the art implementing a predictable variation of the methods and/or systems disclosed in Xpipes 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.

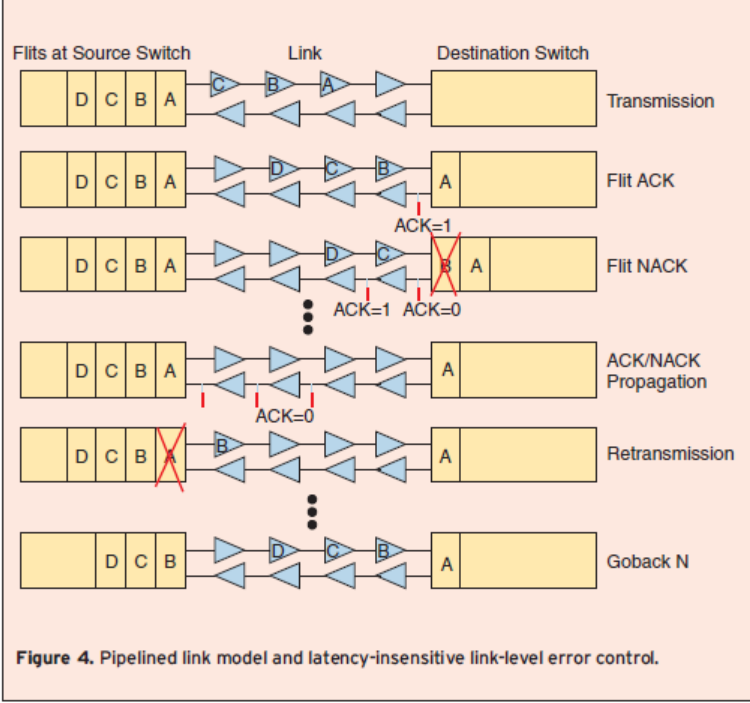
¹ Nothing in these claim charts should be construed as an admission regarding infringement, either literally or under the doctrine of equivalents, or as an admission regarding Defendants' understanding of the proper scope of the Asserted Claim. Given Plaintiff's vague and ambiguous infringement contentions, 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.

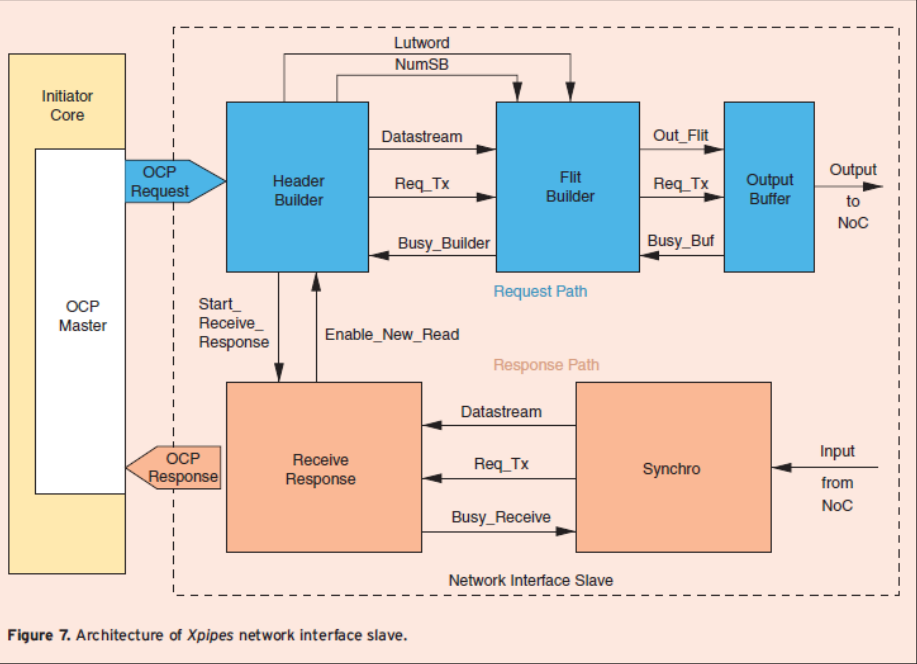
A. Claim 10

'2893 Patent Claim Limitation	Xpipes: A Network-on-Chip Architecture for Gigascale Systems-on-Chips ("Xpipes")
[10pre-a] A method of designing an integrated circuit comprising a plurality of functional blocks, and a data communication network comprising a plurality of network stations being interconnected via a plurality of communication channels for communicating data packages between the functional blocks,	To the extent the preamble is limiting, Xpipes discloses and/or renders obvious this part of the preamble based on at least the following citations. See, e.g.: <i>See, e.g.</i>, p. 18 "The growing complexity of embedded multi-processor architectures for digital media processing will soon require highly scalable communication infrastructures. Packet switched Networks-on-Chip (NoC) have been proposed to support the trend for Systems-on-Chip integration. In this paper, an advanced NoC architecture, called Xpipes, targeting high performance and reliable communication for on-chip multi-processors is introduced. It consists of a library of soft macros (switches, network interfaces and links) that are design-time composable and tunable so that domain-specific heterogeneous architectures can be instantiated and synthesized. Links can be pipelined with a flexible number of stages to decouple link throughput from its length and to get arbitrary topologies. Moreover, a tool called XpipesCompiler, which automatically instantiates a customized NoC from the library of soft network components, is used in this paper to test the Xpipes-based synthesis flow for domain-specific communication architectures." <i>See, e.g.</i>, p. 19: "A scalable communication infrastructure that better supports the trend of SoC integration consists of an on-chip packet-switched micro-network of interconnects, generally known as Network-on-Chip (NoC) architecture [3, 4, 5]. The basic idea is borrowed from traditional large-scale multi-processors and the wide-area networks domain, and envisions on-chip router (or switch)-based networks on which packetized communication takes place, as depicted in Fig. 1. Cores access the network by means of proper interfaces, and have their packets forwarded to destination through a multi-hop routing path." <i>See, e.g.</i>, p. 19: "The scalable and modular nature of NoCs and their support for efficient on-chip communication potentially leads to NoC-based multi-processor systems characterized by high structural complexity and functional diversity." "Figure 1. Example of Network-on-Chip architecture."

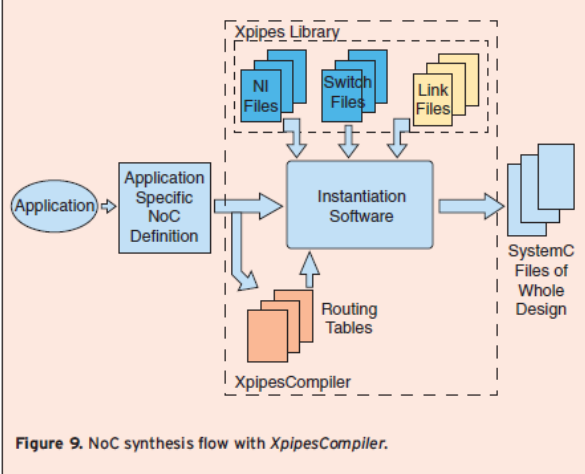
'2893 Patent Claim Limitation	Xpipes: A Network-on-Chip Architecture for Gigascale Systems-on-Chips ("Xpipes")
	 Figure 1. Example of Network-on-Chip architecture. See, e.g., p. 20: “Xpipes has been designed as a library of highly parameterised soft macros (network interface, switch and switch-to-switch link), which are design-time tunable and composable. Therefore, arbitrary topologies can be instantiated and the challenging issue of heterogeneous NoC design can be addressed. In fact, an illustrative example will show how Xpipes can be employed in a design space exploration framework, wherein the ability of several customized NoC topologies to efficiently accommodate the communication requirements of an application is investigated and compared.” See, e.g., p. 22: “Data packetization is a critical task for the network interface, and has an impact on the communication latency, in addition to the latency of the communication channel. Messages that have to be transmitted across the network are usually partitioned into fixed-length packets. Packets in turn are often broken into message flow control units called flits. In the presence of channel width constraints, multiple physical channel cycles can be used to transfer a single flit. A phit is the unit of information that can be transferred across a physical channel in a single step. Flits represent logical

'2893 Patent Claim Limitation	Xpipes: A Network-on-Chip Architecture for Gigascale Systems-on-Chips (“Xpipes”)
	units of information, as opposed to phits that correspond to physical quantities. In many implementations, a flit is set to be equal to a phit. The packet preparation process consists of building the packet header, payload and packet tail. The header contains the necessary routing and network control information (e.g., the source and destination address). When source routing is used, the destination address is ignored and replaced with a route field that specifies the route to destination. This overhead in terms of packet header is counterbalanced by a simpler routing logic at the network switches: they simply have to look at the routing fields in the packets and route them over the specified switch output port. The packet tail indicates the end of a packet and usually contains parity bits for error-detecting or error-correcting codes. Packetizing data deals effectively with communication errors. In fact, packet boundaries contain the effect of errors and allow error recovery on a packet-by-packet basis.” <i>See, e.g., p. 24-25: “A solution to overcome the interconnect-delay problem consists of pipelining interconnects [34, 35]. Wires can be partitioned into segments (or relay stations, which have a function similar to the one of latches in a pipelined data path) whose length satisfies pre-defined timing requirements (e.g., the desired clock speed of the design). In this way, link delay is changed into latency, but the data introduction rate is not bound by the link delay any more. Now the latency of a channel connecting two modules may end up being more than one clock cycle. This requires the system to be composed of modules whose behavior does not depend on the latency of input communication channels [34].</i> Xpipes interconnect makes use of pipelined links and of latency-insensitive operation of its building blocks. Switch-to-switch links are subdivided into basic segments whose length can be tailored to the desired clock frequency that needs to be achieved in the design. In this way, the system operating frequency is not bound by the delay of long links. According to the link length, a certain number of clock cycles is needed by a flit to cross the interconnect. If network switches are designed in such a way that their functional correctness depends on the flit arriving order and not on their timing, the input links of the switches can be different and of any length. These design choices are at the basis of latency insensitive operation of the NoC and allow the construction of an arbitrary network topology and hence support for heterogeneous architectures. Fig. 4 illustrates the link model, which is equivalent to a pipelined shift register. Pipelining has been used both for data and control lines. The figure also illustrates how pipelined links affect

'2893 Patent Claim Limitation	Xpipes: A Network-on-Chip Architecture for Gigascale Systems-on-Chips ("Xpipes")
	the operation of latency-insensitive link-level error control: the retransmission of a corrupted flit between two successive switches is represented. Multiple outstanding flits propagate across the link during the same clock cycle. When flits are correctly received at the destination switch, an ACK is propagated back to the source, and after N clock cycles (where N is the link length expressed in terms of number of repeater stages) the flit will be discarded from the buffer of the source switch. On the contrary, a corrupted flit is NACKed and will be retransmitted in due time." "Figure 4. Pipelined link model and latency-insensitive link-level error control."  Figure 4. Pipelined link model and latency-insensitive link-level error control. See, e.g., p. 27: "The Xpipes network interface provides a standardized OCP interface to networked cores. The NI for cores that initiate communication (initiators) needs to turn OCP-compliant transactions into packets to be transmitted across the network. It represents the slave side of an OCP

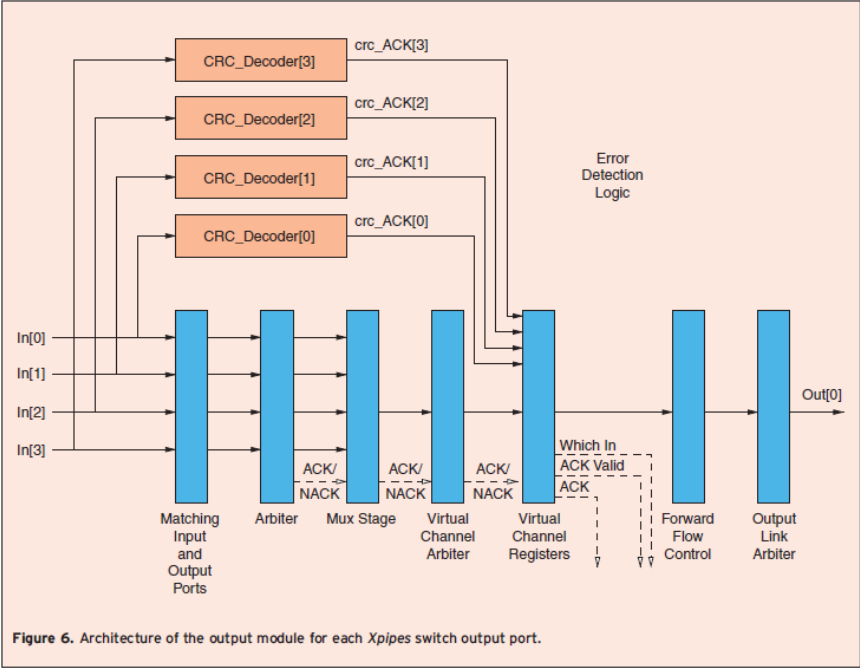
'2893 Patent Claim Limitation	Xpipes: A Network-on-Chip Architecture for Gigascale Systems-on-Chips ("Xpipes")
	end-to-end connection (the master side being the initiator core), and it is therefore referred to as network interface slave (NIS). Its architecture is showed in Fig. 7.” “Figure 7. Architecture of Xpipes network interface slave.”  Figure 7. Architecture of Xpipes network interface slave. See, e.g., p. 28-29: “The individual components of SoCs are inherently heterogeneous with widely varying functionality and communication requirements. The communication infrastructure should optimally match communication patterns among these components. Looking for the most efficient solution can be a (possibly automated) step of a more general design methodology of custom domain-specific NoCs, and Xpipes could be used within this framework as a library of components for the synthesis of the selected NoC configuration. Therefore, for a Xpipes-based design methodology, a tool is needed to automatically instantiate network building blocks (switches, network interfaces and links) from the library of

'2893 Patent Claim Limitation	Xpipes: A Network-on-Chip Architecture for Gigascale Systems-on-Chips (“Xpipes”)
	composable soft macros described in SystemC. XpipesCompiler [32] is the tool designed to automatically instantiate a Xpipes-based application-specific NoC for heterogeneous on-chip multi-processors. The complete XpipesCompiler-assisted NoC design flow is depicted in Fig. 9. From the specification of an application, the designer (or a high-level analysis and exploration tool) creates a high-level view of the SoC floorplan, including nodes (with their network interfaces), links and switches. Based on the clock speed target and link routing, the number of pipeline stages for each link is also determined. The information on the network architecture is then specified in an input file for the XpipesCompiler. Routing tables for the network interfaces are also specified. The tool takes the Xpipes library of soft network components as an additional input. The output is a SystemC hierarchical description, which includes all switches, links, network nodes and interfaces and specifies their connectivity. Then the final description can be compiled and simulated at the cycle-accurate and signal-accurate level. At this point, the description can be fed to backend RTL synthesis tools for silicon implementation. In a nutshell, the XpipesCompiler generates a set of network component instances that are custom-tailored to the specification contained in its input network description file. This tool allows a very instructive comparison of the effects (in terms of area, power and performance) of mapping applications on customized domain-specific NoCs and regular mesh NoCs.” “Figure 9. NoC synthesis flow with XpipesCompiler.”

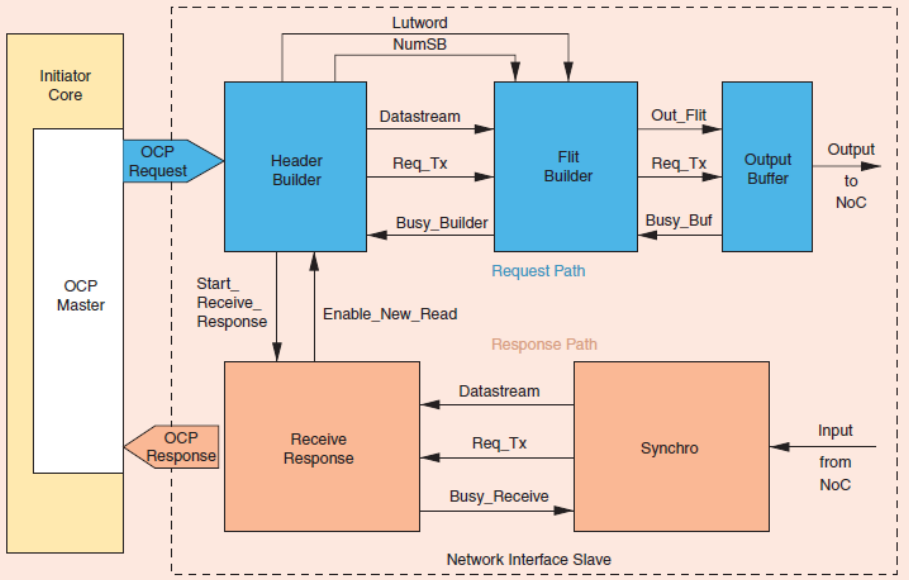
'2893 Patent Claim Limitation	Xpipes: A Network-on-Chip Architecture for Gigascale Systems-on-Chips ("Xpipes")
	 Figure 9. NoC synthesis flow with XpipesCompiler. Furthermore, to the extent this part of the preamble is not disclosed in Xpipes, it (and the claim as a whole) would have been obvious based on the combinations identified in Defendants' Invalidity Contentions.
[10pre-b] each data package comprising N data elements including a data element comprising routing information for the network stations, N being an integer of at least two,	To the extent the preamble is limiting, Xpipes discloses and/or renders obvious this part of the preamble based on at least the following citations. See, e.g.: See, e.g., p. 22: "Data packetization is a critical task for the network interface, and has an impact on the communication latency, in addition to the latency of the communication channel. Messages that have to be transmitted across the network are usually partitioned into fixed-length packets. Packets in turn are often broken into message flow control units called flits. In the presence of channel width constraints, multiple physical channel cycles can be used to transfer a single flit. A phit is the unit of information that can be transferred across a physical channel in a single step. Flits represent logical units of information, as opposed to phits that correspond to physical quantities. In many implementations, a flit is set to be equal to a phit. The packet preparation process consists of building the packet header, payload and packet tail. The header contains the necessary routing and network control information (e.g., the source and

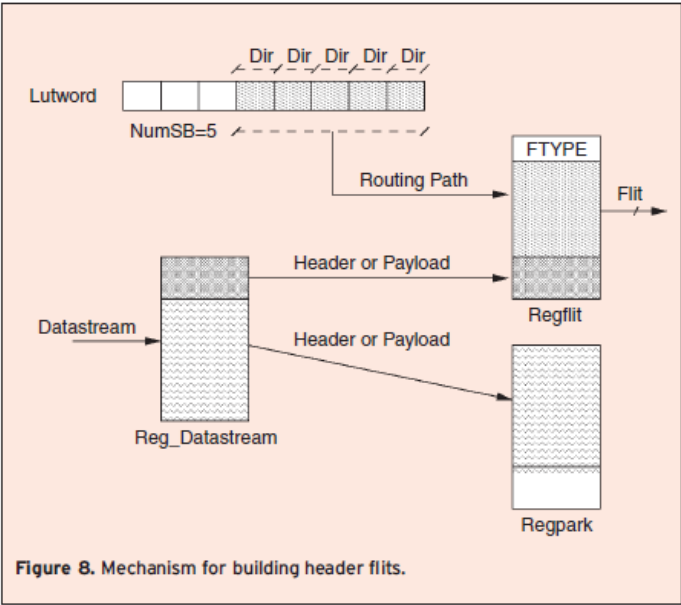
'2893 Patent Claim Limitation	Xpipes: A Network-on-Chip Architecture for Gigascale Systems-on-Chips ("Xpipes")
	destination address). When source routing is used, the destination address is ignored and replaced with a route field that specifies the route to destination. This overhead in terms of packet header is counterbalanced by a simpler routing logic at the network switches: they simply have to look at the routing fields in the packets and route them over the specified switch output port. The packet tail indicates the end of a packet and usually contains parity bits for error-detecting or error-correcting codes. Packetizing data deals effectively with communication errors. In fact, packet boundaries contain the effect of errors and allow error recovery on a packet-by-packet basis." <i>See, e.g.</i>, p. 22: "Finally, a wormhole routing scheme can be employed to reduce switch memory requirements and to permit low latency communication. The first flit of a packet contains routing information, and header flit decoding enables the switches to establish the path and subsequent flits simply follow this path in a pipelined fashion by means of switch output port reservation. A flit is passed to the next switch as soon as enough space is available to store it, even though there is not enough space to store the whole packet. If a certain flit faces a busy channel, subsequent flits have to wait at their current locations and are therefore spread over multiple switches, thus blocking the intermediate links. This scheme avoids buffering the full packet at one switch and keeps end-to-end latency low, although it is more sensitive to deadlock and may result in low link utilization." <i>See, e.g.</i>, p. 23: "Xpipes network interfaces use OCP as point-to-point communication protocol with the cores, and take care of protocol conversion to adapt to the network protocol. Data packetization results in the packet partitioning procedure illustrated in Fig. 3. A flit type field allows to identify the head and the tail flit, and to distinguish between header and payload flits. The NoC backbone relies on a wormhole switching technique and makes use of a static routing algorithm called street sign routing. Routes are derived by the network interface by accessing a look-up table based on the destination address. Such information consists of direction bits read by each switch and indicating the output port of the switch that flits belonging to a certain packet have to be directed to. This routing algorithm allows a lightweight switch implementation as no dynamic decisions have to be taken at the switching nodes." "Figure 3. Packet partitioning into flits"

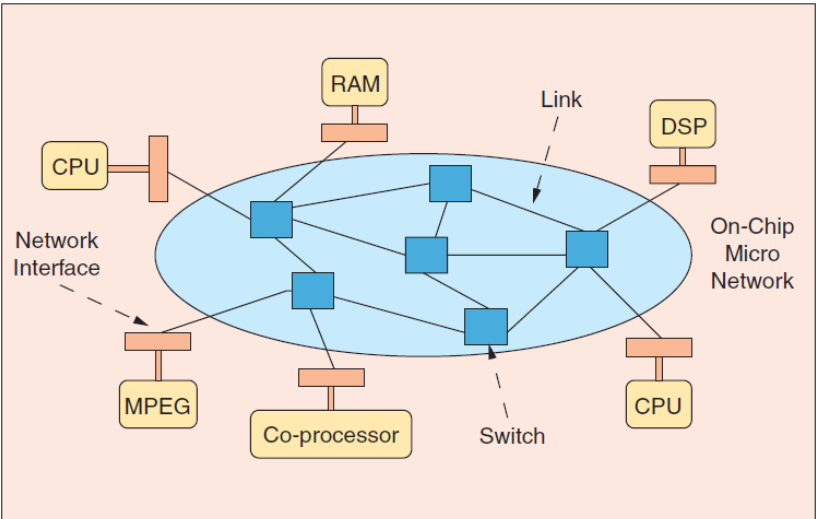
'2893 Patent Claim Limitation	Xpipes: A Network-on-Chip Architecture for Gigascale Systems-on-Chips (“Xpipes”)
	Figure 3. Packet partitioning into flits. See, e.g., p. 27-28: “As can be observed from Fig. 6, each output module is deeply pipelined (7 pipeline stages) so to maximize the operating clock frequency of the switch. The CRC decoders for error detection work in parallel with switch operation, thereby hiding their latency. The first pipeline stage checks the header of incoming packets on the different input ports to determine whether those packets have to be routed through the output port under consideration. Only matching packets are forwarded to the second stage, which resolves contention based on a round-robin policy. Arbitration is carried out when the tail flit of the preceding packet is received, so that all other flits of a packet can be propagated without contention related delay at this stage. A NACK for flits of non-selected packets is generated. The third stage is just a multiplexer, which selects the prioritised input port. The following arbitration stage keeps the status of virtual channel registers and determines whether flits can be stored into the registers or not. A header flit is sent to the register with the greatest number of free locations, and followed by successive flits of the same packet. The fifth stage is the actual buffering stage, and the ACK/NACK response at this stage indicates whether a flit has been successfully stored or not. The following stage takes care of forward flow control: a flit is transmitted to the next switch only when adequate free register locations are available at the output port of interest of the destination switch. Finally, a last

'2893 Patent Claim Limitation	Xpipes: A Network-on-Chip Architecture for Gigascale Systems-on-Chips (“Xpipes”)
	arbitration stage multiplexes the virtual channels on the physical output link on a flit-by-flit basis, thereby improving network throughput.” “Figure 6. Architecture of the output module for each Xpipes switch output port.”  Figure 6. Architecture of the output module for each Xpipes switch output port. See, e.g., p. 27-28: “The Xpipes network interface provides a standardized OCP interface to networked cores. The NI for cores that initiate communication (initiators) needs to turn OCP-compliant transactions into packets to be transmitted across the network. It represents the slave side of an OCP end-to-end connection (the master side being the initiator core), and it is therefore referred to as network interface slave (NIS). Its architecture is showed in Fig. 7. The NIS has to build the packet header, which has to be spread over a variable number of flits depending on the length of the path to the destination node. The lookup table containing static routing information is accessed by the HEADER_BUILDER block and the destination address is

'2893 Patent Claim Limitation	Xpipes: A Network-on-Chip Architecture for Gigascale Systems-on-Chips (“Xpipes”)
	used as table entry. In this way, two routing fields are extracted: numSB (the number of hops to destination) and lutword (the actual direction bits read from the look-up table). Together with the core transaction related information (datastream), they are forwarded to the FLIT_BUILDER block, provided the enable signal busy_builder is not asserted. Based on the input data, the module FLIT_BUILDER has the task of building the flits to be transmitted via the output buffer OUTPUT_BUFFER, according to the mechanism illustrated in Fig. 8. Let us assume that a packet requires numSB = 5 hops to get to the destination, and that the direction to be taken at each switch is expressed by DIR. The module FLIT_BUILDER builds the first flit by concatenating the flit type field with path information. If there is some space left in the flit, it is filled with header information derived from the input datastream. The unused part of the datastream is stored in a regpark register, so that a new datastream can be read from the HEADER_BUILDER block. The following header and/or payload flits will be formed by combining data stored in regpark and reg_datastream. No partially filled flits are transmitted to make transmission more efficient. Finally, the module OUTPUT_BUFFER stores flits to be sent across the network, and allows the NIS to keep preparing successive flits also when the network is congested. Size of this buffer is a design parameter. The response phase is carried out by means of two modules. SYNCHRO receives incoming flits and reads out only useful information (e.g., it discards routing fields). At the same time, it contains buffering resources to synchronize the network’s requests to transmit the remaining flits of a packet with the core consuming rate. The RECEIVE_RESPONSE module translates useful header and payload information into OCP-compliant response fields. When a read transaction is initiated by the initiator core, the HEADER_BUILDER block asserts a start_receive_response signal that triggers the waiting phase of the RECEIVE_RESPONSE module for the requested data. As a consequence, the NIS supports only one outstanding read operation to keep interface complexity low. Although no read after read transactions can be initiated unless the previous one has been completed, an indefinite number of write transactions can be carried out after an outstanding read or write has been initiated. The architecture of a network interface master is similar to the one just described, and is not reported here for lack of space.” “Figure 7. Architecture of Xpipes network interface slave.”

'2893 Patent Claim Limitation	Xpipes: A Network-on-Chip Architecture for Gigascale Systems-on-Chips (“Xpipes”)
	 Figure 7. Architecture of <i>Xpipes</i> network interface slave. “Figure 8. Mechanism for building header flits.”

'2893 Patent Claim Limitation	Xpipes: A Network-on-Chip Architecture for Gigascale Systems-on-Chips (“Xpipes”)
	 Figure 8. Mechanism for building header flits. Furthermore, to the extent this part of the preamble is not disclosed in Xpipes, it (and the claim as a whole) would have been obvious based on the combinations identified in Defendants’ Invalidity Contentions.
[10pre-c] the plurality of network stations comprising a plurality of data routers and a plurality of network interfaces, each of the data routers being coupled to a functional block via a network interface; the method comprising the acts of:	To the extent the preamble is limiting, Xpipes discloses and/or renders obvious this part of the preamble based on at least the following citations. See, e.g.: See, e.g., p. 18 “The growing complexity of embedded multi-processor architectures for digital media processing will soon require highly scalable communication infrastructures. Packet switched Networks-on-Chip (NoC) have been proposed to support the trend for Systems-on-Chip integration. In this paper, an advanced NoC architecture, called Xpipes, targeting high performance and reliable communication for on-chip multi-processors is introduced. It consists of a library of soft macros (switches, network interfaces and links) that are design-time composable and tunable so that domain-specific heterogeneous architectures can be instantiated and synthesized. Links can be pipelined with a flexible number of stages to decouple link throughput from its length and to get

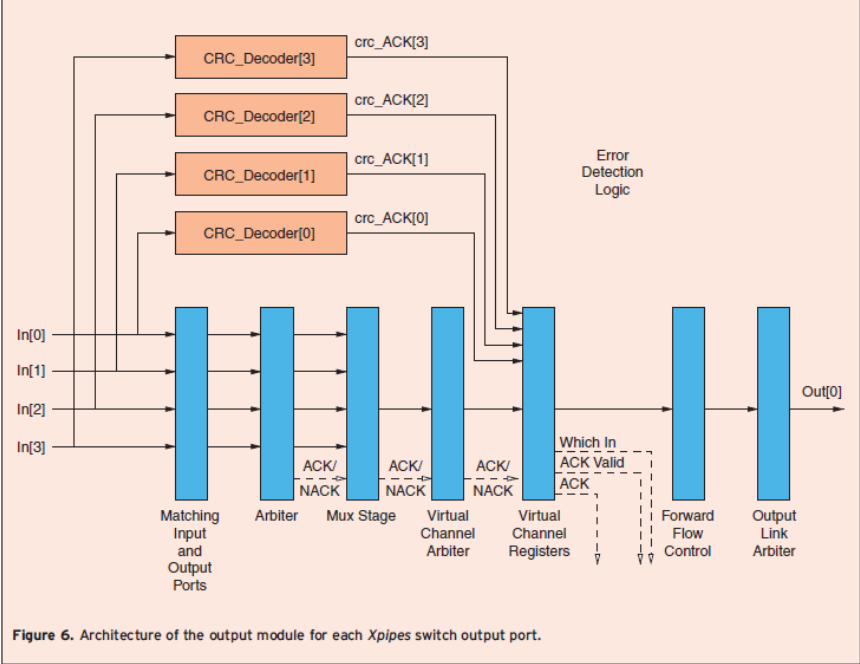
'2893 Patent Claim Limitation	Xpipes: A Network-on-Chip Architecture for Gigascale Systems-on-Chips ("Xpipes")
	arbitrary topologies. Moreover, a tool called XpipesCompiler, which automatically instantiates a customized NoC from the library of soft network components, is used in this paper to test the Xpipes-based synthesis flow for domain-specific communication architectures." See, e.g., p. 19: "A scalable communication infrastructure that better supports the trend of SoC integration consists of an on-chip packet-switched micro-network of interconnects, generally known as Network-on-Chip (NoC) architecture [3, 4, 5]. The basic idea is borrowed from traditional large-scale multi-processors and the wide-area networks domain, and envisions on-chip router (or switch)-based networks on which packetized communication takes place, as depicted in Fig. 1. Cores access the network by means of proper interfaces, and have their packets forwarded to destination through a multi-hop routing path." "Figure 1. Example of Network-on-Chip architecture."  The diagram illustrates a Network-on-Chip (NoC) architecture. At the center is an 'On-Chip Micro Network' represented by a light blue oval. Inside this network, there are several blue square nodes connected by lines, representing routers or switches. A dashed arrow labeled 'Link' points to one of these internal connections. Surrounding the central network are various functional blocks: a 'CPU' at the top left, 'RAM' at the top, a 'DSP' at the top right, a 'CPU' at the bottom right, a 'Co-processor' at the bottom, and an 'MPEG' block at the bottom left. Each of these blocks is connected to the central network via a 'Network Interface', which is depicted as a small orange rectangle with a T-shaped connection point. A dashed arrow labeled 'Network Interface' points to the interface of the MPEG block. Below the diagram, the caption 'Figure 1. Example of Network-on-Chip architecture.' is repeated. Figure 1. Example of Network-on-Chip architecture.

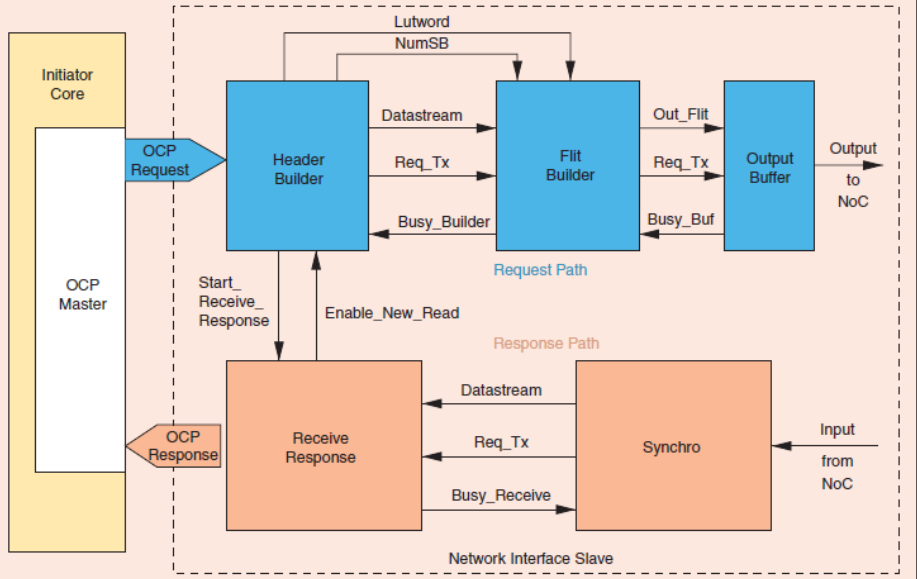
'2893 Patent Claim Limitation	Xpipes: A Network-on-Chip Architecture for Gigascale Systems-on-Chips (“Xpipes”)
	See, e.g., p. 20: “Xpipes has been designed as a library of highly parameterised soft macros (network interface, switch and switch-to-switch link), which are design-time tunable and composable. Therefore, arbitrary topologies can be instantiated and the challenging issue of heterogeneous NoC design can be addressed. In fact, an illustrative example will show how Xpipes can be employed in a design space exploration framework, wherein the ability of several customized NoC topologies to efficiently accommodate the communication requirements of an application is investigated and compared.” See, e.g., p. 21: “Finally, let us observe that NoC components (e.g., switches or interfaces) can be instantiated multiple times in the same design (as opposed to the arbiter of traditional shared busses, which is instance-specific) and reused in a large number of products targeting a specific application domain.” See, e.g., p. 21-22: “In general, the most relevant tasks of a <i>network interface (NI)</i> are: (i) hiding the details about network communication protocol to the cores, so that they can be developed independently of the communication infrastructure, (ii) communication protocol conversion (from end-to-end to network protocol), (iii) data packetization (packet assembly, delivery and disassembly). The former objective can be achieved by means of standard interfaces, such as VCI [21] and OCP [22], whose main characteristics are a high degree of configurability to adapt to the core’s functionality and the independence of the request and response phases, thus supporting multiple outstanding requests and pipelining of transfers.” See, e.g., p. 22: “Data packetization is a critical task for the network interface, and has an impact on the communication latency, in addition to the latency of the communication channel. Messages that have to be transmitted across the network are usually partitioned into fixed-length packets. Packets in turn are often broken into message flow control units called flits. In the presence of channel width constraints, multiple physical channel cycles can be used to transfer a single flit. A phit is the unit of information that can be transferred across a physical channel in a single step. Flits represent logical units of information, as opposed to phits that correspond to physical quantities. In many implementations, a flit is set to be equal to a phit.

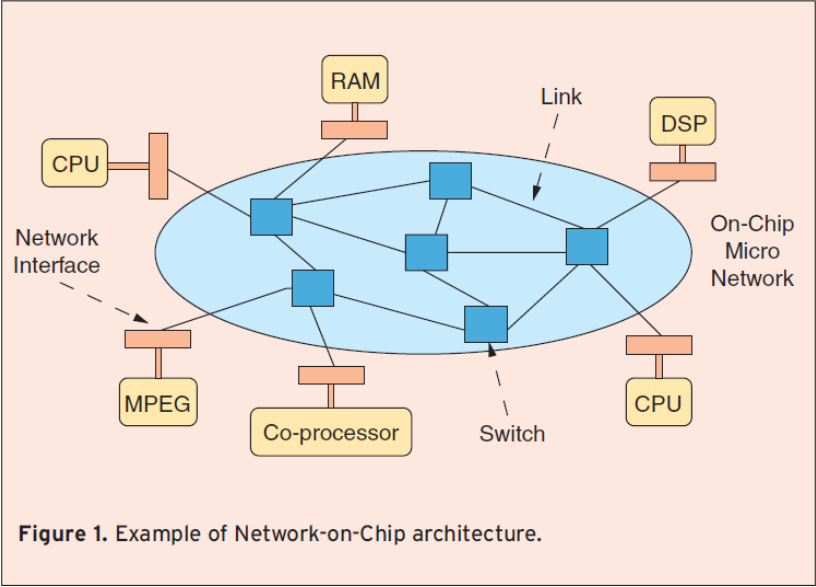
'2893 Patent Claim Limitation	Xpipes: A Network-on-Chip Architecture for Gigascale Systems-on-Chips ("Xpipes")
	The packet preparation process consists of building the packet header, payload and packet tail. The header contains the necessary routing and network control information (e.g., the source and destination address). When source routing is used, the destination address is ignored and replaced with a route field that specifies the route to destination. This overhead in terms of packet header is counterbalanced by a simpler routing logic at the network switches: they simply have to look at the routing fields in the packets and route them over the specified switch output port. The packet tail indicates the end of a packet and usually contains parity bits for error-detecting or error-correcting codes. Packetizing data deals effectively with communication errors. In fact, packet boundaries contain the effect of errors and allow error recovery on a packet-by-packet basis." <i>See, e.g.</i>, p. 22: "The task of the switch is to carry packets injected into the network to their final destination, following a statically defined or dynamically determined routing path. The switch transfers packets from one of its input ports to one or more of its output ports." <i>See, e.g.</i>, p. 24-25: "A solution to overcome the interconnect-delay problem consists of pipelining interconnects [34, 35]. Wires can be partitioned into segments (or relay stations, which have a function similar to the one of latches in a pipelined data path) whose length satisfies pre-defined timing requirements (e.g., the desired clock speed of the design). In this way, link delay is changed into latency, but the data introduction rate is not bound by the link delay any more. Now the latency of a channel connecting two modules may end up being more than one clock cycle. This requires the system to be composed of modules whose behavior does not depend on the latency of input communication channels [34]. Xpipes interconnect makes use of pipelined links and of latency-insensitive operation of its building blocks. Switch-to-switch links are subdivided into basic segments whose length can be tailored to the desired clock frequency that needs to be achieved in the design. In this way, the system operating frequency is not bound by the delay of long links. According to the link length, a certain number of clock cycles is needed by a flit to cross the interconnect. If network switches are designed in such a way that their functional correctness depends on the flit arriving order and not on their timing, the input links of the switches can be different and of any length. These design choices are at the basis of latency insensitive operation of the NoC and allow the construction of an arbitrary network topology and hence support for heterogeneous architectures.

'2893 Patent Claim Limitation	Xpipes: A Network-on-Chip Architecture for Gigascale Systems-on-Chips ("Xpipes")
	Fig. 4 illustrates the link model, which is equivalent to a pipelined shift register. Pipelining has been used both for data and control lines. The figure also illustrates how pipelined links affect the operation of latency-insensitive link-level error control: the retransmission of a corrupted flit between two successive switches is represented. Multiple outstanding flits propagate across the link during the same clock cycle. When flits are correctly received at the destination switch, an ACK is propagated back to the source, and after N clock cycles (where N is the link length expressed in terms of number of repeater stages) the flit will be discarded from the buffer of the source switch. On the contrary, a corrupted flit is NACKed and will be retransmitted in due time." "Figure 4. Pipelined link model and latency-insensitive link-level error control." Figure 4. Pipelined link model and latency-insensitive link-level error control.

'2893 Patent Claim Limitation	Xpipes: A Network-on-Chip Architecture for Gigascale Systems-on-Chips (“Xpipes”)
	See, e.g., p.27: “As can be observed from Fig. 6, each output module is deeply pipelined (7 pipeline stages) so to maximize the operating clock frequency of the switch. The CRC decoders for error detection work in parallel with switch operation, thereby hiding their latency. The first pipeline stage checks the header of incoming packets on the different input ports to determine whether those packets have to be routed through the output port under consideration. Only matching packets are forwarded to the second stage, which resolves contention based on a round-robin policy. Arbitration is carried out when the tail flit of the preceding packet is received, so that all other flits of a packet can be propagated without contention related delay at this stage. A NACK for flits of non-selected packets is generated. The third stage is just a multiplexer, which selects the prioritised input port. The following arbitration stage keeps the status of virtual channel registers and determines whether flits can be stored into the registers or not. A header flit is sent to the register with the greatest number of free locations, and followed by successive flits of the same packet. The fifth stage is the actual buffering stage, and the ACK/NACK response at this stage indicates whether a flit has been successfully stored or not. The following stage takes care of forward flow control: a flit is transmitted to the next switch only when adequate free register locations are available at the output port of interest of the destination switch. Finally, a last arbitration stage multiplexes the virtual channels on the physical output link on a flit-by-flit basis, thereby improving network throughput.” “Figure 6. Architecture of the output module for each Xpipes switch output port.”

'2893 Patent Claim Limitation	Xpipes: A Network-on-Chip Architecture for Gigascale Systems-on-Chips (“Xpipes”)
	 Figure 6. Architecture of the output module for each Xpipes switch output port.
	See, e.g., p. 27: “The Xpipes network interface provides a standardized OCP interface to networked cores. The NI for cores that initiate communication (initiators) needs to turn OCP-compliant transactions into packets to be transmitted across the network. It represents the slave side of an OCP end-to-end connection (the master side being the initiator core), and it is therefore referred to as network interface slave (NIS). Its architecture is showed in Fig. 7.” “Figure 7. Architecture of Xpipes network interface slave.”

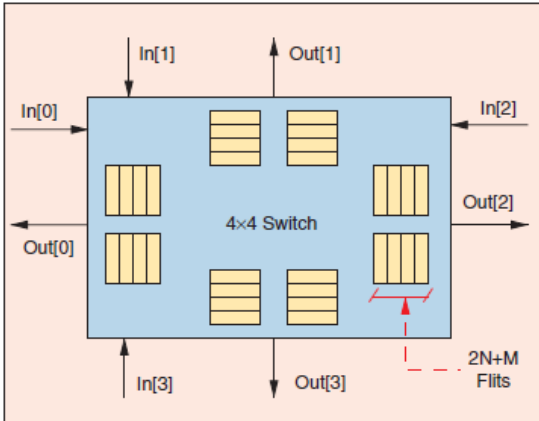
'2893 Patent Claim Limitation	Xpipes: A Network-on-Chip Architecture for Gigascale Systems-on-Chips (“Xpipes”)
	 Figure 7. Architecture of Xpipes network interface slave. Furthermore, to the extent this part of the preamble is not disclosed in Xpipes, it (and the claim as a whole) would have been obvious based on the combinations identified in Defendants’ Invalidity Contentions.
[10a] identifying a first communication channel between a first network station and a second network station that has a data transfer delay exceeding a predefined delay threshold; and	Xpipes discloses and/or renders obvious this limitation based on at least the following citations. See, e.g.: <i>See, e.g.</i>, p. 18 “The growing complexity of embedded multi-processor architectures for digital media processing will soon require highly scalable communication infrastructures. Packet switched Networks-on-Chip (NoC) have been proposed to support the trend for Systems-on-Chip integration. In this paper, an advanced NoC architecture, called Xpipes, targeting high performance and reliable communication for on-chip multi-processors is introduced. It consists of a library of soft macros (switches, network interfaces and links) that are design-time composable and tunable so that

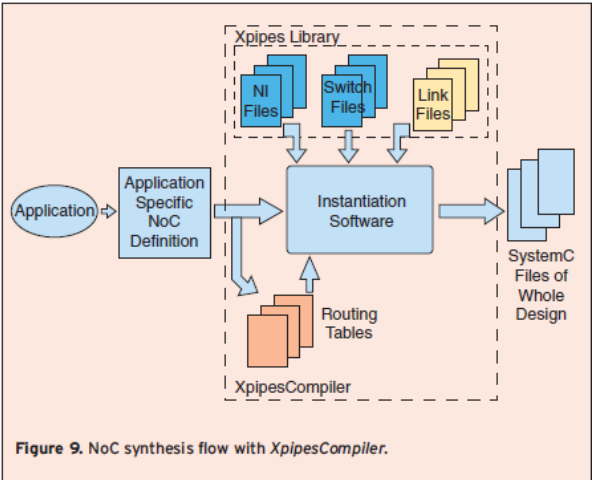
'2893 Patent Claim Limitation	Xpipes: A Network-on-Chip Architecture for Gigascale Systems-on-Chips ("Xpipes")
	domain-specific heterogeneous architectures can be instantiated and synthesized. Links can be pipelined with a flexible number of stages to decouple link throughput from its length and to get arbitrary topologies. Moreover, a tool called XpipesCompiler, which automatically instantiates a customized NoC from the library of soft network components, is used in this paper to test the Xpipes-based synthesis flow for domain-specific communication architectures." "Figure 1. Example of Network-on-Chip architecture."  Figure 1. Example of Network-on-Chip architecture. See, e.g., p. 20: "Technology issues. While gate delays scale down with technology, global wire delays typically increase or remain constant as repeaters are inserted. It is estimated that in 50 nm technology, at a clock frequency of 10 GHz, a global wire delay can be up to 6–10 clock cycles [3]. Therefore, limiting the on-chip distance travelled by critical signals will be key to guarantee the performance of the overall system, and will be a common design guideline for all kinds of system interconnects.

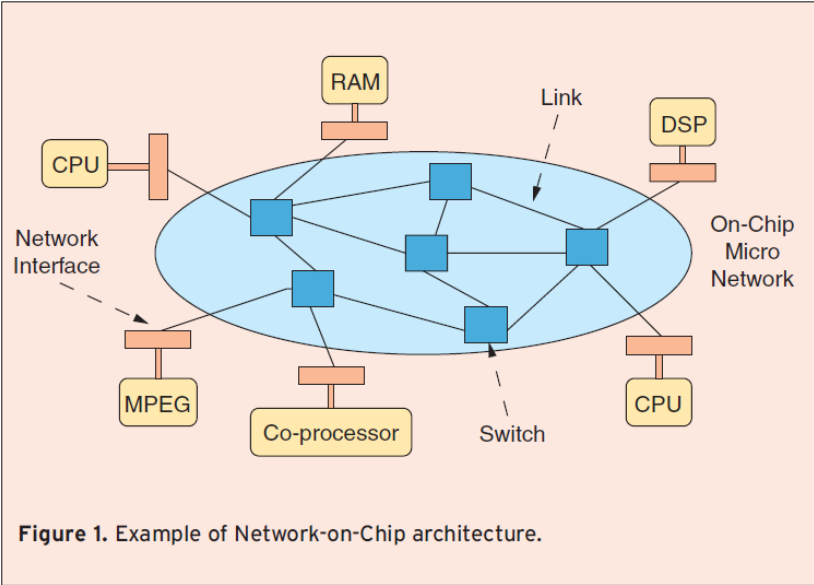
'2893 Patent Claim Limitation	Xpipes: A Network-on-Chip Architecture for Gigascale Systems-on-Chips (“Xpipes”)
	Synchronization of cores on future SoCs will be unfeasible due to deep submicron effects (clock skew, power associated with clock distribution trees, etc.), and an alternative scenario consists of self-synchronous cores that communicate with one another through a network-centric architecture [14]. Finally, signal integrity issues (cross-talk, power supply noise, soft errors, etc.) will lead to more transient and permanent failures of signals, logic values, devices and interconnects, thus raising the reliability concern for on-chip communication [15]. In many cases, on-chip networks can be designed as regular structures, allowing electrical parameters of wires to be optimised and well controlled. This leads to lower communication failure probabilities, thus enabling the use of low swing signalling techniques [16], and to the capability of exploiting performance optimisation techniques such as wavefront pipelining [17].” <i>See, e.g.,</i> p. 23: “Finally, the design of communication (switch-to-switch and network interface-to-switch) links represents a critical issue with respect to the system performance. As geometries shrink, gate delay improves much faster than the delay in long wires. Therefore, the long wires increasingly determine the maximum clock rate, and hence performance, of the entire design. The problem becomes particularly serious for domain-specific heterogeneous SoCs, where the wire structure is highly irregular and may include both short and extremely long switch-to-switch links. Techniques have to be devised to decouple link throughput from its length and to allow functionally correct operation of the switches in presence of mis-aligned inputs due to the unequal length of the input links.” <i>See, e.g.,</i> p. 24-25: “A solution to overcome the interconnect-delay problem consists of pipelining interconnects [34, 35]. Wires can be partitioned into segments (or relay stations, which have a function similar to the one of latches in a pipelined data path) whose length satisfies pre-defined timing requirements (e.g., the desired clock speed of the design). In this way, link delay is changed into latency, but the data introduction rate is not bound by the link delay any more. Now the latency of a channel connecting two modules may end up being more than one clock cycle. This requires the system to be composed of modules whose behavior does not depend on the latency of input communication channels [34].

'2893 Patent Claim Limitation	Xpipes: A Network-on-Chip Architecture for Gigascale Systems-on-Chips (“Xpipes”)
	Xpipes interconnect makes use of pipelined links and of latency-insensitive operation of its building blocks. Switch-to-switch links are subdivided into basic segments whose length can be tailored to the desired clock frequency that needs to be achieved in the design. In this way, the system operating frequency is not bound by the delay of long links. According to the link length, a certain number of clock cycles is needed by a flit to cross the interconnect. If network switches are designed in such a way that their functional correctness depends on the flit arriving order and not on their timing, the input links of the switches can be different and of any length. These design choices are at the basis of latency insensitive operation of the NoC and allow the construction of an arbitrary network topology and hence support for heterogeneous architectures. Fig. 4 illustrates the link model, which is equivalent to a pipelined shift register. Pipelining has been used both for data and control lines. The figure also illustrates how pipelined links affect the operation of latency-insensitive link-level error control: the retransmission of a corrupted flit between two successive switches is represented. Multiple outstanding flits propagate across the link during the same clock cycle. When flits are correctly received at the destination switch, an ACK is propagated back to the source, and after N clock cycles (where N is the link length expressed in terms of number of repeater stages) the flit will be discarded from the buffer of the source switch. On the contrary, a corrupted flit is NACKed and will be retransmitted in due time.” “Figure 4. Pipelined link model and latency-insensitive link-level error control.”

'2893 Patent Claim Limitation	Xpipes: A Network-on-Chip Architecture for Gigascale Systems-on-Chips (“Xpipes”)
	Figure 4. Pipelined link model and latency-insensitive link-level error control. See, e.g., p. 25-26: “A schematic representation of a switch for the Xpipes NoC is illustrated in Fig. 5. The example configuration exhibits 4 inputs, 4 outputs and 2 virtual channels multiplexed across the same physical output link. Switch operation is latency insensitive, in that correct operation is guaranteed for arbitrary link pipeline depth. For latency insensitive operation, the switch has virtual channel registers to store $2N + M$ flits, where N is the link length (expressed as number of basic repeater stages) and M is a switch architecture related contribution (12 cycles in this design). The reason is that each transmitted flit has to be acknowledged before being discarded from the buffer. Before an ACK is received, the flit has to propagate across the link (N cycles), an ACK/NACK decision has to be taken at the destination switch (a portion of M cycles), the ACK/NACK signal has to be propagated back (N cycles) and recognized by the source switch (remaining portion of M cycles). During this time, other $2N + M$ flits are transmitted but not yet ACKed.”

'2893 Patent Claim Limitation	Xpipes: A Network-on-Chip Architecture for Gigascale Systems-on-Chips (“Xpipes”)
	“Figure 5. Example of a 4 x 4 switch configuration with 2 virtual channels”  Figure 5. Example of a 4 x 4 switch configuration with 2 virtual channels. See, e.g., p. 28-29: “The individual components of SoCs are inherently heterogeneous with widely varying functionality and communication requirements. The communication infrastructure should optimally match communication patterns among these components. Looking for the most efficient solution can be a (possibly automated) step of a more general design methodology of custom domain-specific NoCs, and Xpipes could be used within this framework as a library of components for the synthesis of the selected NoC configuration. Therefore, for a Xpipes-based design methodology, a tool is needed to automatically instantiate network building blocks (switches, network interfaces and links) from the library of composable soft macros described in SystemC. XpipesCompiler [32] is the tool designed to automatically instantiate a Xpipes-based application-specific NoC for heterogeneous on-chip multi-processors. The complete XpipesCompiler-assisted NoC design flow is depicted in Fig. 9. From the specification of an application, the designer (or a high-level analysis and exploration tool) creates a high-level view of the SoC floorplan, including nodes (with their network interfaces), links and switches. Based on the clock speed target and link routing, the number of pipeline stages for each

'2893 Patent Claim Limitation	Xpipes: A Network-on-Chip Architecture for Gigascale Systems-on-Chips (“Xpipes”)
	link is also determined. The information on the network architecture is then specified in an input file for the XpipesCompiler. Routing tables for the network interfaces are also specified. The tool takes the Xpipes library of soft network components as an additional input. The output is a SystemC hierarchical description, which includes all switches, links, network nodes and interfaces and specifies their connectivity. Then the final description can be compiled and simulated at the cycle-accurate and signal-accurate level. At this point, the description can be fed to backend RTL synthesis tools for silicon implementation. In a nutshell, the XpipesCompiler generates a set of network component instances that are custom-tailored to the specification contained in its input network description file. This tool allows a very instructive comparison of the effects (in terms of area, power and performance) of mapping applications on customized domain-specific NoCs and regular mesh NoCs.” “Figure 9. NoC synthesis flow with XpipesCompiler.”  Figure 9. NoC synthesis flow with XpipesCompiler. Furthermore, to the extent this limitation is not disclosed in Xpipes, this limitation (and the claim as a whole) would have been obvious based on the combinations identified in Defendants’ Invalidity Contentions.

'2893 Patent Claim Limitation	Xpipes: A Network-on-Chip Architecture for Gigascale Systems-on-Chips ("Xpipes")
[10b] in response to the identifying act, inserting $M*N$ data storage elements into the data communication network, M being a positive integer, for introducing a delay of $M*N$ cycles on the first communication channel.	Xpipes discloses and/or renders obvious this limitation based on at least the following citations. See, e.g.: “Figure 1. Example of Network-on-Chip architecture.”  Figure 1. Example of Network-on-Chip architecture. See, e.g., p. 20: “Technology issues. While gate delays scale down with technology, global wire delays typically increase or remain constant as repeaters are inserted. It is estimated that in 50 nm technology, at a clock frequency of 10 GHz, a global wire delay can be up to 6–10 clock cycles [3]. Therefore, limiting the on-chip distance travelled by critical signals will be key to guarantee the performance of the overall system, and will be a common design guideline for all kinds of system interconnects. Synchronization of cores on future SoCs will be unfeasible due to deep submicron effects (clock skew, power associated with clock distribution trees, etc.), and an alternative scenario consists of self-synchronous cores that communicate with one another through a network-centric architecture [14]. Finally, signal integrity issues (cross-talk, power supply noise, soft errors, etc.) will lead to more transient and permanent failures of signals, logic values, devices and interconnects,

'2893 Patent Claim Limitation	Xpipes: A Network-on-Chip Architecture for Gigascale Systems-on-Chips (“Xpipes”)
	thus raising the reliability concern for on-chip communication [15]. In many cases, on-chip networks can be designed as regular structures, allowing electrical parameters of wires to be optimised and well controlled. This leads to lower communication failure probabilities, thus enabling the use of low swing signalling techniques [16], and to the capability of exploiting performance optimisation techniques such as wavefront pipelining [17].” <i>See, e.g.</i>, p. 22: “The switch design is usually characterized by a power-performance trade-off: power-hungry switch memory resources can be required by the need to support high performance on-chip communication. A specific design of a switch may include both input and output buffers or only one type of buffers. Input queuing uses fewer buffers, but suffers from head-of-line blocking. Virtual output queuing has a higher performance, but at the cost of more buffers. Network flow control (or routing mode) specifically addresses the limited amount of buffering resources in switches. Three policies are feasible in this context [33]. In store-and-forward routing, an entire packet is received and entirely stored before being forwarded to the next switch. This is the most demanding approach in terms of memory requirements and switch latency. Also virtual cut-through routing requires buffer space for an entire packet, but allows lower latency communication, in that a packet is forwarded as soon as the next switch guarantees that the complete packet will be accepted. If this is not the case, the current router must be able to store the whole packet. Finally, a wormhole routing scheme can be employed to reduce switch memory requirements and to permit low latency communication. The first flit of a packet contains routing information, and header flit decoding enables the switches to establish the path and subsequent flits simply follow this path in a pipelined fashion by means of switch output port reservation. A flit is passed to the next switch as soon as enough space is available to store it, even though there is not enough space to store the whole packet. If a certain flit faces a busy channel, subsequent flits have to wait at their current locations and are therefore spread over multiple switches, thus blocking the intermediate links. This scheme avoids buffering the full packet at one switch and keeps end-to-end latency low, although it is more sensitive to deadlock and may result in low link utilization.” <i>See, e.g.</i>, p. 23: “Finally, the design of communication (switch-to-switch and network interface-to-switch) links represents a critical issue with respect to the system performance. As geometries

'2893 Patent Claim Limitation	Xpipes: A Network-on-Chip Architecture for Gigascale Systems-on-Chips ("Xpipes")
	shrink, gate delay improves much faster than the delay in long wires. Therefore, the long wires increasingly determine the maximum clock rate, and hence performance, of the entire design. The problem becomes particularly serious for domain-specific heterogeneous SoCs, where the wire structure is highly irregular and may include both short and extremely long switch-to-switch links. Techniques have to be devised to decouple link throughput from its length and to allow functionally correct operation of the switches in presence of mis-aligned inputs due to the unequal length of the input links." <i>See, e.g.</i>, p. 23: "The high degree of parameterisation of Xpipes network building blocks regards both global network-specific parameters and block-specific parameters. The former ones include flit size, degree of redundancy of error control logic, address space of the cores, maximum number of hops between any two nodes, maximum number of bits for end-to-end flow control, etc. On the contrary, parameters specific to the network interface are: type of interface (master, slave or both), flit buffer size at the output port, content of routing tables for source-based routing, other interface parameters to the cores such as number of address/data lines, maximum burst length, etc. Parameterisation of the switches mainly regard the number of I/O ports, the number of virtual channels for each physical output link and the link buffer size. Finally, the length of each individual link can be specified in terms of number of repeater stages, as will be discussed next." <i>See, e.g.</i>, p. 24-25: "A solution to overcome the interconnect-delay problem consists of pipelining interconnects [34, 35]. Wires can be partitioned into segments (or relay stations, which have a function similar to the one of latches in a pipelined data path) whose length satisfies pre-defined timing requirements (e.g., the desired clock speed of the design). In this way, link delay is changed into latency, but the data introduction rate is not bound by the link delay any more. Now the latency of a channel connecting two modules may end up being more than one clock cycle. This requires the system to be composed of modules whose behavior does not depend on the latency of input communication channels [34]. Xpipes interconnect makes use of pipelined links and of latency-insensitive operation of its building blocks. Switch-to-switch links are subdivided into basic segments whose length can be tailored to the desired clock frequency that needs to be achieved in the design. In this way, the system operating frequency is not bound by the delay of long links. According to the link length, a

'2893 Patent Claim Limitation	Xpipes: A Network-on-Chip Architecture for Gigascale Systems-on-Chips (“Xpipes”)
	certain number of clock cycles is needed by a flit to cross the interconnect. If network switches are designed in such a way that their functional correctness depends on the flit arriving order and not on their timing, the input links of the switches can be different and of any length. These design choices are at the basis of latency insensitive operation of the NoC and allow the construction of an arbitrary network topology and hence support for heterogeneous architectures. Fig. 4 illustrates the link model, which is equivalent to a pipelined shift register. Pipelining has been used both for data and control lines. The figure also illustrates how pipelined links affect the operation of latency-insensitive link-level error control: the retransmission of a corrupted flit between two successive switches is represented. Multiple outstanding flits propagate across the link during the same clock cycle. When flits are correctly received at the destination switch, an ACK is propagated back to the source, and after N clock cycles (where N is the link length expressed in terms of number of repeater stages) the flit will be discarded from the buffer of the source switch. On the contrary, a corrupted flit is NACKed and will be retransmitted in due time.” “Figure 4. Pipelined link model and latency-insensitive link-level error control.”

'2893 Patent Claim Limitation	Xpipes: A Network-on-Chip Architecture for Gigascale Systems-on-Chips (“Xpipes”)
	Figure 4. Pipelined link model and latency-insensitive link-level error control. Furthermore, to the extent this limitation is not disclosed in Xpipes, this limitation (and the claim as a whole) would have been obvious based on the combinations identified in Defendants’ Invalidity Contentions.