This document serves solely as a demonstration of AlphaClaim's technology and does not advocate legal or administrative action against any entity, including entities mentioned herein. This document neither alleges nor implies that any service or product from these entities infringes, either directly or indirectly, upon any patent or other intellectual property rights, including patents mentioned herein. This document neither alleges nor implies the invalidity or unenforceability of any patent, including patents mentioned herein.

This document was created independently by PriceWire, Inc. using AlphaClaim, for illustrative purposes only, without any third-party compensation or under any client contract or direction.

PriceWire, Inc. makes no representations or warranties regarding the completeness or accuracy of the information contained in this document and expressly disclaims any liability related to the use of such information for any purpose. PriceWire, Inc. is not a law firm, and this document does not provide legal advice.

US10355975B2 Case Study Report

This case study report reviews how AlphaClaim was used to automatically claim chart 2,500 references in 8 hours to find a high-quality invalidity ground (a well-motivated combination of three references) for claim 13 of US10355975B2.

Section	Page #
What is AlphaClaim?	3
How does AlphaClaim work?	4
Case & Invalidity Ground Overview	5
Reviewed Grounds Scores	6
AlphaScore Cross-reference Analytics	6
Generated Claim Chart	7-12

What is AlphaClaim?

AlphaClaim is a “brain” that computes accurate and detailed claim charts for a variety of document types against a given set of claims, adhering to a preponderance of the evidence standard. AlphaClaim has been aligned on 1000s of PTAB IPR institution and final written decisions.

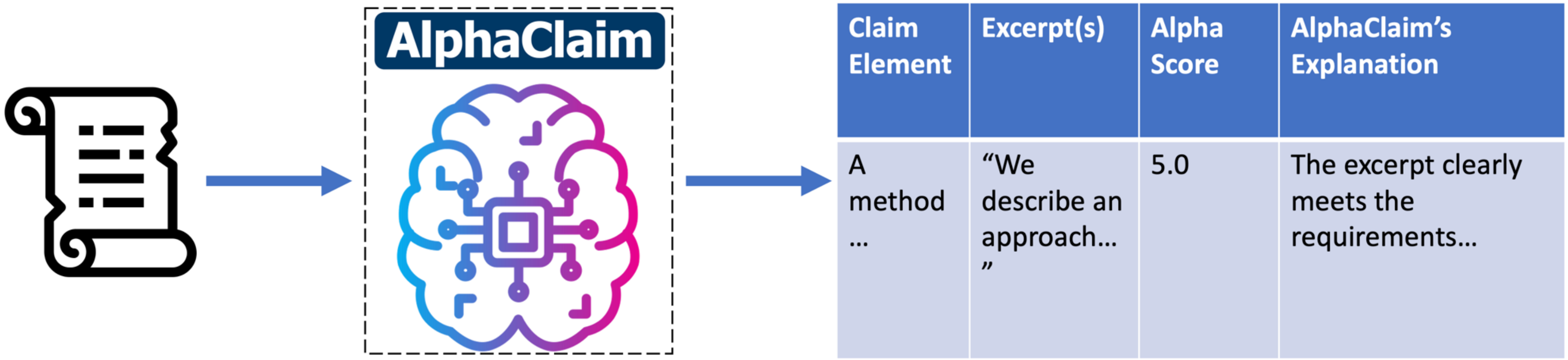
AlphaClaim performs automated, accurate, exhaustive claim charting of **superhuman quantities** of documents (often >10,000). For every document, AlphaClaim produces a claim chart with excerpts and explanations. It then computes an “AlphaScore” out of 5 for each claim element, indicating the strength of the document’s disclosure of that element.

AlphaClaim can be applied in three ways. This document focuses on the first.

(1) To **identify the best invalidity grounds** (including combinations), for IPRs.
An AlphaScore of 4 or 5 indicates evidence stronger than the median IPR petition.

(2) To **identify evidence of patent validity** prior to assertion or sale of a patent.
If AlphaClaim’s exhaustive review turns up low AlphaScores for all documents, a patent owner can be more confident that the patent will survive IPR or other 102/103 challenges.

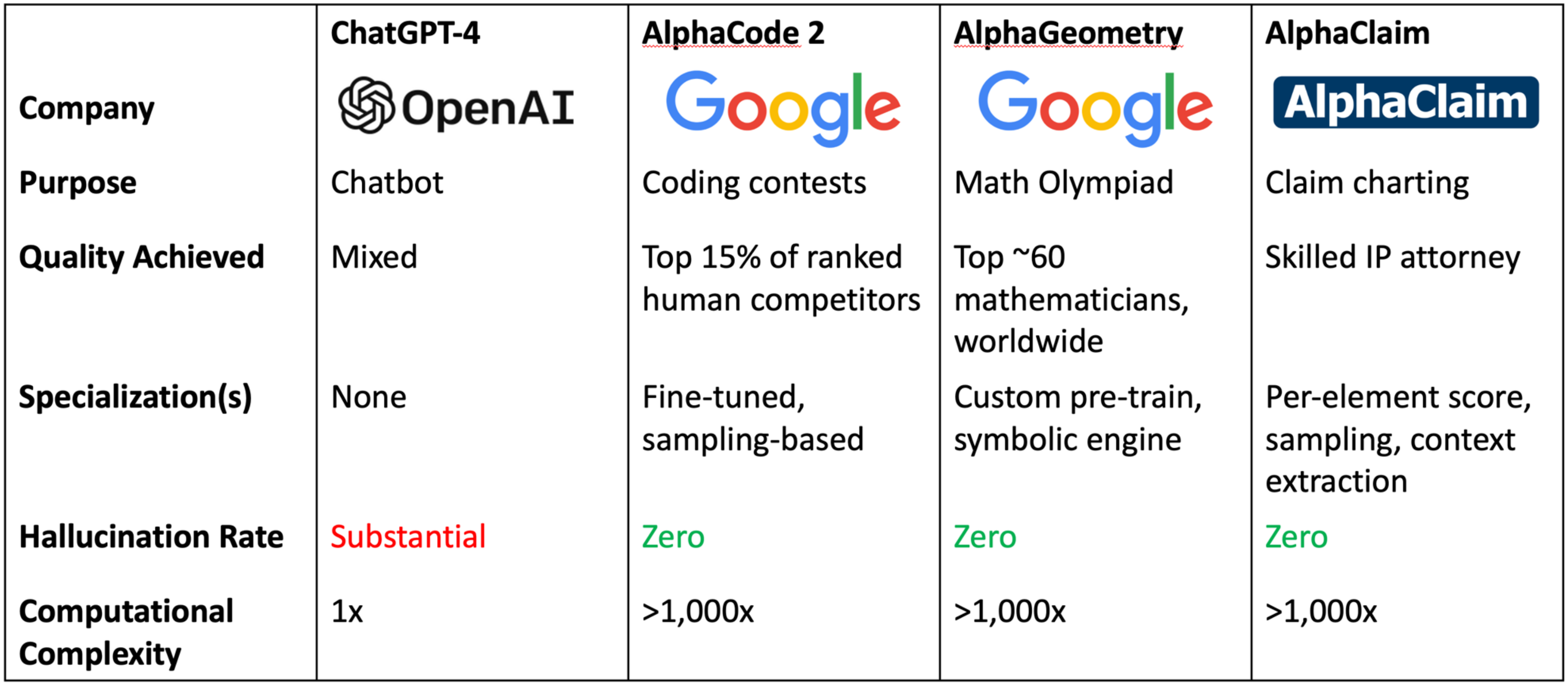
(3) To **identify the best evidence of infringement**, for patent owners.
An AlphaScore of 4 or 5 indicates evidence stronger than the median filed claim chart.



Claim Element	Excerpt(s)	Alpha Score	AlphaClaim’s Explanation
A method …	“We describe an approach… ”	5.0	The excerpt clearly meets the requirements…

How does AlphaClaim work?

AlphaClaim leverages many state-of-the-art AI technologies that are used in systems that achieve quality equivalent to the best humans, as detailed in the chart below. While computationally more expensive than consumer-grade chatbots, AlphaClaim achieves high quality with zero hallucination.



	ChatGPT-4	AlphaCode 2	AlphaGeometry	AlphaClaim
Company	OpenAI	Google	Google	AlphaClaim
Purpose	Chatbot	Coding contests	Math Olympiad	Claim charting
Quality Achieved	Mixed	Top 15% of ranked human competitors	Top ~60 mathematicians, worldwide	Skilled IP attorney
Specialization(s)	None	Fine-tuned, sampling-based	Custom pre-train, symbolic engine	Per-element score, sampling, context extraction
Hallucination Rate	Substantial	Zero	Zero	Zero
Computational Complexity	1x	>1,000x	>1,000x	>1,000x

AlphaClaim has 8 patents pending for its technological innovations.

Case & Invalidity Ground Overview

In this report, we show how in a matter of hours, AlphaClaim was able to find a high-quality invalidity ground for claim 13 of the '975 patent. **AlphaClaim's three-reference ground achieves an AlphaScore of 4.3/5, stronger than the median IPR petition.**

Patent-in-suit (US10355975B2)

Originally issued to and still owned by Rex Computing with a priority date of 10/19/2016. The patent covers the multiprocessor SoC architecture of the Rex chip, including its cacheless design. Has been asserted against Cerebras Systems, with the case in active litigation (trial scheduled for Sept. 2024). Cerebras filed an IPR against multiple claims of the '975 patent, with 6 invalidity grounds. The PTAB's Final Written Decision found none of the challenged claims unpatentable.

How AlphaClaim was used

We used AlphaClaim to automatically, accurately, exhaustively claim chart approximately 2,500 prior art references. First, we provided AlphaClaim a claim construction, in technical language, based on the claim constructions adopted by the court in the related case. This took about 15 minutes. The main AlphaClaim process, claim charting prior art, then took about 8 hours. AlphaClaim charted all the references we provided it, after it removed grace-period prior art and art already cited during prosecution. We used AlphaClaim analytics to break up the claim elements and find a combination of references that rendered the claim obvious. AlphaClaim also provides a "MotivScore", which estimates motivation to combine references through multiple factors, including whether the art is analogous, relative overlap, and more. By ranking all charted references based on their AlphaScores, AlphaClaim was able to find a high-scoring (4.3/5) three-reference invalidity ground.

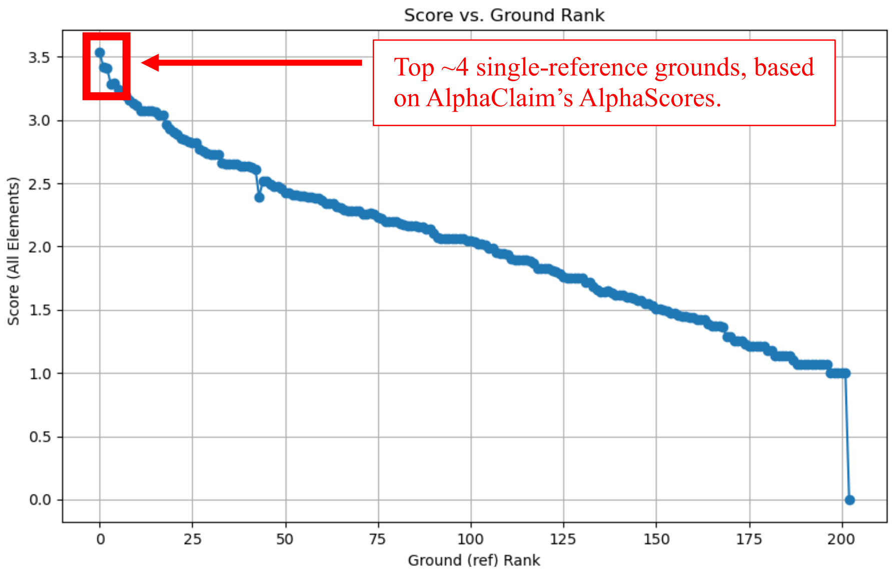
The ground identified

AlphaClaim chose the Epiphany processor architecture reference document as its base reference (available since at least July 2014). For the static priority and output port FIFOs, AlphaClaim relied on the conference paper *Networks on Chip: A Synthesis Perspective* (published in 2005 conference). Finally, for the optimization module, we kept one original MIT Raw reference, as the PTAB did not find issue with the mapping for ground 1. This three-reference ground fixes two problems the PTAB identified with Cerebras's ground 1 (for claim 13): (1) no presence of scratchpad memories; and (2) no FIFOs present on NoC router output ports.

To provide more insight into how AlphaClaim works, we show a score graph for the top 200 references AlphaClaim charted and the AlphaScore cross-reference analytics which helped us decide how to break up the claim elements for combination. We then provide the unedited claim chart, AlphaScores, and explanations generated by AlphaClaim for its three-reference ground.

Reviewed Grounds Scores

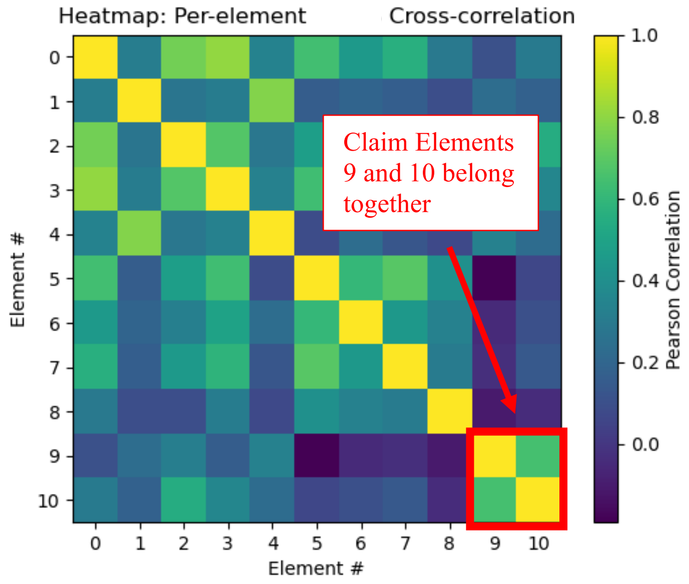
AlphaClaim works by automatically, accurately, and exhaustively claim charting the prior art provided by the user to find the best invalidity ground possible. The chart shows the results of individual per-reference AlphaScores for the single-reference grounds reviewed in this particular case. While AlphaClaim reviewed ~2,500 references for this case, we terminate the chart at around 200 documents for viewability, as scores drop too low below that. The per-reference AlphaScore is computed as a mean of the per-element AlphaScores within each reference.



Top ~4 single-reference grounds, based on AlphaClaim's AlphaScores.

AlphaScore Cross-reference Analytics

Some claims can be anticipated with a single reference. **In this case, the chart above – showing the single-reference scores topping out at around 3.5 – indicates that we need a multi-reference combination to have a better chance before the PTAB; we recommend a score of 4 to 5**. We achieved this by combining three references, as suggested by AlphaClaim's "Per-Element Cross Correlation" chart. By charting and scoring ~2,500 references, AlphaClaim has "discovered" which claim elements belong together, and which do not.



Claim Elements 9 and 10 belong together

AlphaClaim: Generated Claim Chart

This is the unedited output from AlphaClaim. Any legal document (e.g., IPR petition) would be drafted by a professional based on this output.

Note: the "claim construction" referenced by AlphaClaim is user-provided. This took about 15 minutes to create.

Claim Element (US10355975B2 claim 21)	Reference: AlphaClaim's Extracted Excerpt	Alpha-Score	AlphaClaim's Explanation
A network-on-chip microprocessor chip	**Epiphany Reference:** The eMesh Network-on-Chip consists of three separate and orthogonal mesh structures, each serving different types of transaction traffic: cMesh: Used for write transactions destined for an on-chip mesh node. The cMesh network connects a mesh node to all four of its neighbors and has a maximum bidirectional throughput of 8 bytes/cycle in each of the four routing directions. At an operating frequency of 1GHz, the cMesh network has a total throughput of more than 0.5 Terabit/sec. rMesh: Used for all read requests. The rMesh network connects a mesh node to all four of its neighbors and has a maximum throughput of 1 read transaction every 8 clock cycles in each routing direction. xMesh: Used for write transactions destined for off-chip resources and for passing through transactions destined for another chip in a multi-chip system configuration. The xMesh network allows an array of chips to be connected in a mesh structure without glue logic. The xMesh network is split into a south-north network and an east-west network. The maximum throughput of the mesh depends on the available-off chip I/O bandwidth. Current silicon versions of the Epiphany architecture can sustain a total off-chip bandwidth of 8GB/sec.	5.0	The Excerpt precisely matches the claim construction by describing an instance of a chip (within the context of the Epiphany architecture) that contains a network-on-chip (the eMesh Network-on-Chip). It details the network-on-chip's structure, consisting of three separate mesh structures (cMesh, rMesh, xMesh), each serving different transaction traffic types. This directly aligns with the requirement to detail the presence of a network-on-chip within the chip architecture. The Excerpt goes further to describe the connectivity (to all four neighbors), the purpose of each mesh (write transactions for cMesh and xMesh, read requests for rMesh), and their throughput capabilities, providing a comprehensive understanding of the network-on-chip's functionality. The Excerpt does not deviate from the claim construction's requirements. It explicitly states the presence of a network-on-chip (the eMesh Network-on-Chip) within a chip (as part of the Epiphany architecture), fulfilling the primary requirement. The detailed descriptions of cMesh, rMesh, and xMesh align with the need to indicate the chip's capability for on-chip communication, further solidifying the match.
a set of scratchpad memory modules	**Epiphany Reference:** Each mesh node has a local, aliased, range of memory That is accessible by the mesh node itself starting at address 0x0 and ending at address 0x00007FFF.	4.0	1. The Excerpt mentions a "local, aliased, range of memory" that is accessible by the mesh node itself, which aligns with the requirement of being part of a mesh node. 2. The memory is described as having a specific address range, implying it is readable and writable through software instructions, thus addressable. 3. The Excerpt does not explicitly state that this memory does not act as a cache. However, there is no indication that it functions as a cache either. 4. The term "scratchpad" is not used in the Excerpt, but based on the claim construction, this is not a strict requirement as long as the memory meets the described criteria. The Excerpt matches the claim construction closely but lacks explicit

Claim Element (US10355975B2 claim 21)	Reference: AlphaClaim's Extracted Excerpt	Alpha-Score	AlphaClaim's Explanation
			clarification that the described memory does not act as a cache. This detail is crucial to fully satisfy the claim construction.
a set of tiles arranged in a grid configuration	**Epiphany Reference:** The Epiphany architecture defines a multicore, scalable, shared-memory, parallel computing fabric. It consists of a 2D array of compute nodes connected by a low-latency mesh network-on-chip. Figure 1 shows an implementation of the architecture, highlighting the key components: A superscalar, floating-point RISC CPU in each mesh node that can execute two floating point operations and a 64-bit memory load operation on every clock cycle. Local memory in each mesh node that provides 32 Bytes/cycle of sustained bandwidth and is part of a distributed, shared memory system. Multicore communication infrastructure in each node that includes a network interface, a multi-channel DMA engine, multicore address decoder, and network-monitor. A 2D mesh network that supports on-chip node-to-node communication latencies in nanoseconds, with zero startup overhead.	5.0	The Excerpt precisely matches the claim construction in several ways: 1. It explicitly states that the architecture consists of a 2D array of compute nodes, which aligns with the requirement of having multiple tiles, cores, mesh nodes, or processing elements arranged in a grid. 2. The mention of a low-latency mesh network-on-chip as the connection between these compute nodes further confirms the grid arrangement, fulfilling the architectural layout requirement. 3. The description includes specific components within each mesh node (e.g., a superscalar, floating-point RISC CPU, local memory, multicore communication infrastructure), which aligns with the claim construction that the chip should contain multiple processing elements. 4. The Excerpt does not infer anything beyond what is explicitly stated and provides a clear depiction of the Epiphany architecture as per the claim construction. Given the above analysis, the Excerpt precisely matches all the requirements set forth by the claim construction. There are no differences in wording or content that would detract from a perfect match.
each tile including a processor core and a router communicatively coupled with one another	**Epiphany Reference:** Processor Node Overview Figure 10 shows the components at each processor node, which include: an eCore RISC CPU, multi-bank local memory, multicore-optimized DMA engine, event monitor, and network interface. The node connects to the Epiphany eMesh network through the network interface, a single point of access.	4.0	The claim construction focuses on the requirement for an explicit statement about the connectivity between individual components (tiles, cores, mesh nodes, or processing elements) and a network-on-chip router. The Excerpt mentions that each processor node includes several components and connects to the Epiphany eMesh network through a network interface, which acts as a single point of access. This description aligns with the claim construction by indicating that there is a connection between the processor node (which can be understood as encompassing tiles, cores, or processing elements) and the network-on-chip (the eMesh network) via the network interface. However, the Excerpt does not explicitly use the term "network-on-chip router," but rather refers to a "network interface" as the point of connection to the eMesh network. This difference in terminology is the only minor deviation from the claim construction.
each processor core corresponds to and is communicatively coupled with a different scratchpad	**Epiphany Reference:** Each mesh node has a local, aliased, range of memory That is accessible by the mesh node itself starting at address 0x0 and ending at address 0x00007FFF.	4.0	1. The Excerpt mentions a "local, aliased, range of memory" that is accessible by the mesh node, which suggests a one-to-one association between the mesh node and this memory range. This aligns with the claim construction's requirement for a one-to-one association. 2. The Excerpt specifies that this memory is accessible

Claim Element (US10355975B2 claim 21)	Reference: AlphaClaim's Extracted Excerpt	Alpha-Score	AlphaClaim's Explanation
memory module of the set of scratchpad memory modules			by the mesh node, implying it is addressable and thus readable and writable through software instructions. This meets the claim construction's criteria for scratchpad memory. 3. The Excerpt does not explicitly state that this memory does not act as a cache. However, given that it is described as a "local, aliased, range of memory" accessible for reading and writing, it implies a function more akin to scratchpad memory than cache memory, which aligns with the claim construction. 4. The term "scratchpad" is not used in the Excerpt, but based on the claim construction, this is not a requirement as long as the described memory meets the defined characteristics of scratchpad memory.
each router includes a set of input ports and a set of output ports	**NoC Synthesis:** Our test topology comprises 30 NIs, equally split among initiators and targets, and 15 switches, of which three having 6 input and 6 output ports (those in the center of the mesh), eight 5x5 (at the sides), and four 4x4 (in the corners). Each switch is connected to one initiator and one target NI.	4.0	The claim construction requires the Excerpt to explicitly state that each router includes multiple input and output ports. The Excerpt, while not using the term "router" explicitly, refers to "switches" within a network- on-chip topology, which perform a similar role to routers in directing data traffic. It details the configuration of these switches, including the number of input and output ports for switches located in different parts of the mesh. This information directly addresses the claim construction's requirement by illustrating that these switches (akin to routers in this context) indeed have multiple input and output ports. However, the Excerpt does not explicitly state that "each router (where the router is part of a network- on-chip) includes multiple input ports and multiple output ports" using the exact terminology. Instead, it provides detailed configurations of switches, which serve a similar purpose within the described topology. The Excerpt's focus on switches, their connection to NIs, and the specific configurations based on their location within the mesh, while closely related, does not use the term "router" directly.
wherein each output port includes a FIFO memory element operable to store a data packet for subsequent sending to a router of an adjacent tile	**NoC Synthesis:** in ×pipes, output buffering was chosen, i.e. FIFOs are present on each output port.	5.0	The claim construction mandates that a FIFO must be present in each output port of a router within a NoC. The excerpt directly addresses this requirement by stating that "output buffering was chosen, i.e., FIFOs are present on each output port." This statement confirms that the design choice made in ×pipes involves placing FIFOs at each output port, which is exactly what the claim construction specifies. There is no deviation from the claim construction in terms of the placement and presence of FIFOs at the output ports. The excerpt does not infer or imply but explicitly states the alignment with the claim construction. Therefore, there are no areas where the excerpt differs from the claim construction

Claim Element (US10355975B 2 claim 21)	Reference: AlphaClaim's Extracted Excerpt	Alpha-Score	AlphaClaim's Explanation
			regarding the placement of FIFOs in the output ports of a router in a NoC.
based on a physical destination address of a data packet, each router is operable to send one or more data packets to routers of one or more adjacent tiles or the processor core corresponding to the router	**Epiphany Reference:** The upper 12 bits of the destination address are used to route transactions to their destination. Along the way, these upper bits—6 bits for row and 6 bits for column—are compared to the row-column ID of each mesh node in the routing path. Transactions are routed east if the destination address column tag is less than the column ID of the current router node, and they are routed west if the destination-address column tag is greater than the column ID of the current router node. ... n. When the destination-address column tag matches the mesh-node column ID, a similar comparison is made in the row direction to determine whether the transaction should be routed to the south or to the north. The transaction routing continues until both the row tag and column tag for the destination match the row and column ID of the current mesh node. Then, the transaction is routed into the network interface of mesh node.	5.0	The Excerpt closely matches the claim construction by describing a routing process that uses a fixed destination address (the upper 12 bits for row and column) to make routing decisions within a network-on-chip. It explicitly states how the routing decisions are made by comparing the destination address with the router's (mesh node's) current position (row- column ID) to determine the direction (east, west, south, north) in which the transaction should be routed. This process continues until the transaction reaches the mesh node whose row and column IDs match the destination address's row and column tags, at which point the transaction is routed into the network interface of the mesh node. The Excerpt precisely matches the claim construction's requirement that the router uses a physical destination address to make routing decisions. It details the mechanism of comparison between the destination address and the current router node's position, which is essential for understanding how routing decisions are made in a network-on-chip.
each router implements a static priority routing policy in the event of a traffic condition	**NoC Synthesis:** All instances have been synthesized, placed and eventually routed to provide figures as accurate as possible. Figures 4(a) and 4(b) show trends when varying the amount of I/O ports; since the bulk of the switch logic and buffering in pipes is associated with output ports, which is compounded by a mild dependence on inputs, the area scales up a bit more than linearly with the amount of output ports. Increasing numbers of input ports make arbitration and multiplexing more complex, which results in a 10% worse frequency when moving from four to six inputs. Figures 4(c) and 4(d) depict performance when varying the flit width of packets. When moving from 16 to 38 bits (which is an optimal Flit size for performance once the decomposition of packets into flits is taken into account), a huge area penalty of 64% can be observed. Such penalty is however weakly proportional to flit width, which increased by 138%. This result is logical, since the area for the datapath (including buffers) has to scale linearly with flit width, but arbitration and control logic are unaffected. The maximum operating frequency is also almost unaffected by flit width, which suggests the worsening of wiring congestion to be not critical. The impact of buffering is investigated in Figures 4(e) and 4(f). Since buffering resources represent a significant percentage of the component area, doubling the buffer depth results in a noticeable 54% area penalty.	4.0	The Excerpt does mention arbitration policies, specifically comparing the implementation of a fair round-robin arbitration policy with a baseline fixed priority one. This directly addresses the claim construction's requirement for the Excerpt to state that routers implement fixed priority arbitration. However, the Excerpt does not explicitly state that "each router" implements fixed priority arbitration; it mentions the implementation of a fair round-robin policy "instead of the baseline fixed priority one," implying that fixed priority arbitration is the default or baseline approach. This suggests that routers do use fixed priority arbitration, but it does not explicitly confirm that every router does so.

Claim Element (US10355975B2 claim 21)	Reference: AlphaClaim's Extracted Excerpt	Alpha-Score	AlphaClaim's Explanation
	Also, due to the FIFO nature of the buffers, increased logic and wiring complexity impacts maximum frequency by as much as 52 MHz (around 6%). Implementing a fair round-robin arbitration policy instead of the baseline fixed priority one incurs a noticeable cost. Additional logic to track the status of input ports results in 15% worse area and maximum operating frequency.		
an optimization module configured to: determine optimal function assignment configurations for groups of tiles of the set of tiles	**Raw MIT2002:** For structured applications with a lot of pipelined parallelism or heavy data movement like that found in software circuits, careful orchestration and layout of operations and network routes provided us with maximal performance because it maximizes pertile performance. For these applications (and for the operating system code), we developed a version of the Gnu C compiler that lets the programmer specify the code and communication on a per-tile basis. Although this seems laborious, the alternative for these sorts of performance-oriented applications is an ASIC implementation, which is considerably more work than programming Raw. We are currently working on a new compiler that automates this mode of programming.	3.0	The excerpt explicitly mentions the development of a compiler that enables programmers to specify code and communication on a per-tile basis, which aligns with the claim construction's requirement for a software module that determines the mapping of a program onto hardware elements (tiles, in this case). This approach is aimed at maximizing performance, which is consistent with the claim construction's emphasis on improving performance through specific allocation of program functions. However, the excerpt does not explicitly state the process of allocating particular functions of a program to specific sets of tiles, cores, or processing elements as the claim construction requires. Instead, it focuses on the outcome of such allocation (maximal per-tile performance) and the development of tools to facilitate this process.
assign two or more functions, which communicate at least unilaterally more frequently with one another than with other functions, to groups of adjacent tiles based on an optimal function assignment configuration determination, wherein the two or more functions are assigned to groups of tiles communicatively coupled in square configurations when the function executes	**Raw MIT2002:** The Raw operating system allows both space and time multiplexing of processes. Thus, not only can a Raw processor run multiple independent processes simultaneously, it can context switch them in and out as on a conventional processor. The operating system allocates a rectangular-shaped number of tiles (corresponding to physical threads that can themselves be virtualized) proportional to the amount of computation that is required by that process. When the operating system context-switches in a given process, it finds a contiguous region of tiles that corresponds to the dimension of the process, and resumes the execution of the physical threads. ... on. We assign operations to tiles in a manner that minimizes congestion and then configure the network routes between these operations. This is very much like the process of designing a customized hardware circuit. In Raw, the compiler performs this customization.	5.0	1. The Excerpt implicitly mentions a software module (the Raw operating system and compiler) that determines the mapping of a program onto a set of tiles (physical threads that can be virtualized), which aligns with the first requirement of the claim construction. 2. The Excerpt explicitly states the goal of minimizing congestion (which can be interpreted as an effort to minimize communication time or distance) when assigning operations to tiles. This aligns with the second requirement of the claim construction. 3. The Excerpt clearly states that the operating system allocates a rectangular-shaped number of tiles for processes and finds a contiguous region of tiles for process execution. This precisely matches the third requirement of the claim construction.

We generally recommend that the AlphaScore for every element be a 4 or 5, indicating quality above average IPR petitions.

In this instance, we retained the excerpt and mapping filed by the petitioner, Cerebras, as the PTAB did not find fault with it for this ground.

Claim Element (US10355975B2 claim 21)	Reference: AlphaClaim’s Extracted Excerpt	Alpha-Score	AlphaClaim’s Explanation
optimally when executed by the groups of tiles communicatively coupled in the square configurations and are assigned to groups of tiles communicatively coupled in linear configurations when the function executes optimally when executed by the groups of tiles communicatively coupled in the linear configurations		🟩	