Exhibit B-4: "Architectural Adaptation for Application-Specific Locality Optimizations," Xingbin Zhang et al., International Conference on Computer Design VLSI in Computers and Processors, 1997 ("Zhang")

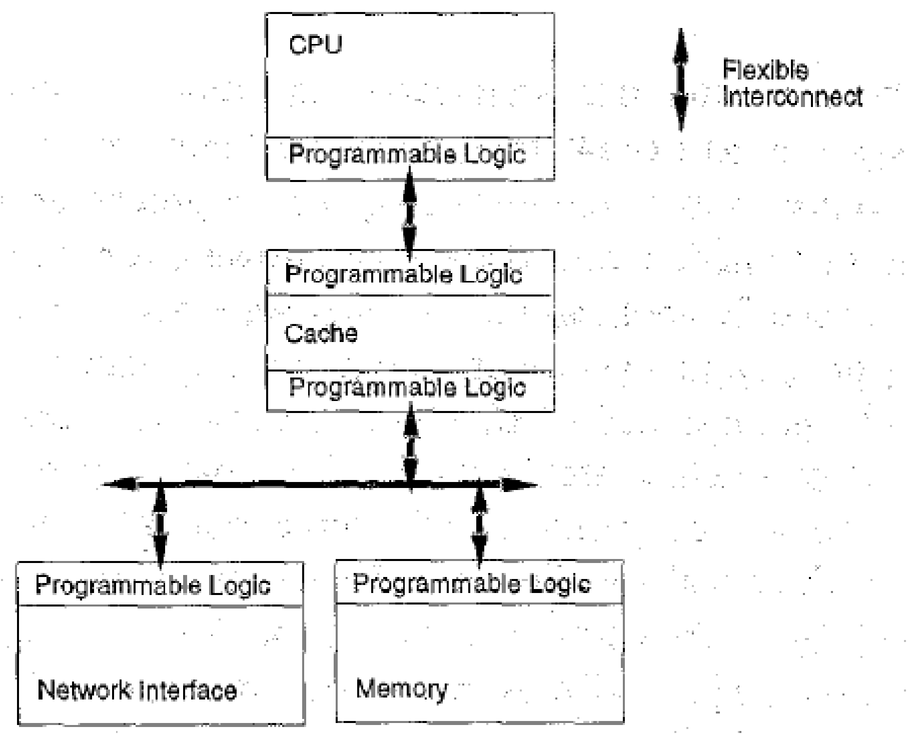
As described in the following claim chart, the asserted claims of the '867 patent are invalid in view of "Architectural Adaptation for Application-Specific Locality Optimizations," Xingbin Zhang et al., International Conference on Computer Design VLSI in Computers and Processors, 1997 ("Zhang").

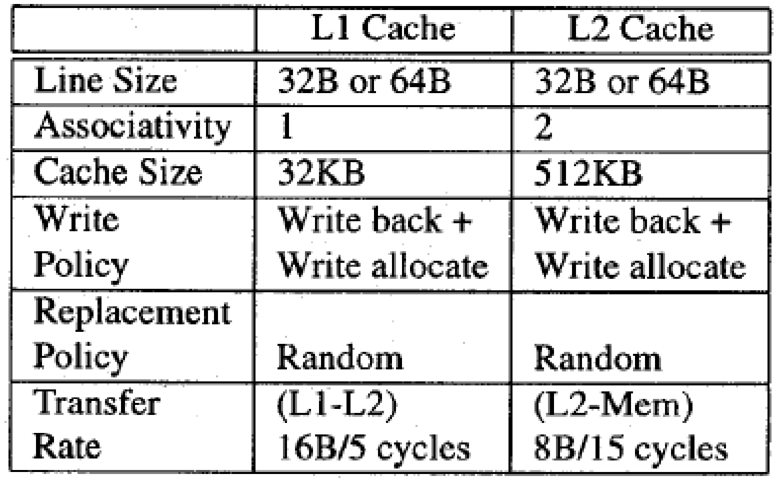
The citations presented herein are exemplary and not exclusive; each prior art reference as a whole discloses each and every limitation of the claims. A citation to a figure or figure reference numeral incorporates by reference the discussion and/or explication of such figure or feature/component referenced by the reference numeral. Further, the mapping in this chart is based on Amazon's present understanding of Plaintiffs' interpretation of the asserted claims of the patent-in-suit as reflected in Plaintiffs' infringement contentions. Nothing in the chart should be regarded as necessarily reflecting how the prior art references would apply to claim elements of the asserted patent under a proper interpretation of the claims. Disclosures cited for dependent claims incorporate by reference the disclosure included herein for the corresponding independent claim.

'867 patent claim 1	**Zhang**
A reconfigurable processor that instantiates an algorithm as hardware comprising:	Zhang discloses a reconfigurable processor that instantiates an algorithm as hardware. For example: **Zhang discloses an architecture with small blocks of programmable logic (e.g., a reconfigurable processor) that can match an application, e.g., performing sparse matrix computations (e.g., instantiate an algorithm as hardware).** Page 151 ("We propose an architecture that integrates small blocks of programmable logic into key elements of a baseline architecture, including processing elements, components of the memory hierarchy, and the scalable interconnect, to provide architectural adaptation - the customization of architectural mechanisms and policies to match an application. … Using sparse matrix computations as examples, our results show that customization for application-specific optimizations can bring significant performance improvement (10X reduction in miss rates, 100X reduction in data traffic), and that an application-driven machine customization provides a promising approach to achieve robust, high performance.")

a first memory having a first characteristic memory bandwidth and/or memory utilization;	Zhang discloses a first memory having a first characteristic memory bandwidth and/or memory utilization. For example: **Zhang discloses cache memories, including an L1 cache which has a transfer rate of 16B/5 cycles (e.g., a characteristic memory bandwidth).** Page 152 ("As an example of its flexibility, MORPH could be used to implement either a cache-coherent machine, a non-cache coherent machine, or even clusters of cache coherent machines connected by put/get or message passing. In this paper, we focus on architectural adaptation in the memory system for locality optimizations such as latency tolerance."); page 153 ("Figure 4 shows the prefetcher implementation using programmable logic integrated with the L1 cache.")

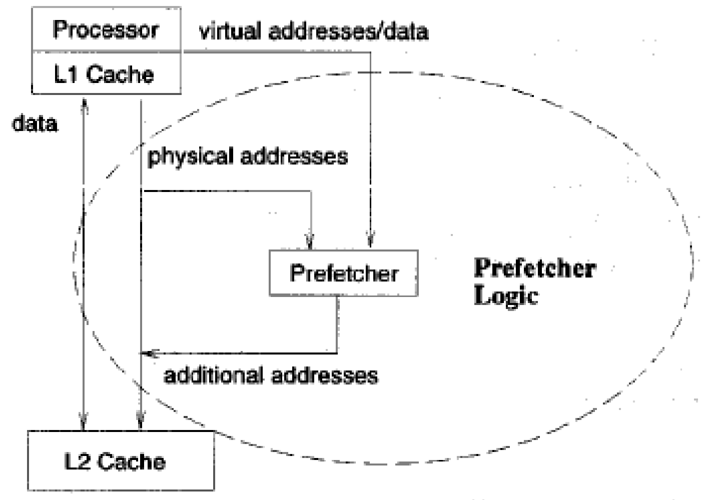
Exhibit B-4: "Architectural Adaptation for Application-Specific Locality Optimizations," Xingbin Zhang et al., International Conference on Computer Design VLSI in Computers and Processors, 1997 ("Zhang")

	 CPU Programmable Logic Flexible Interconnect Programmable Logic Cache Programmable Logic Programmable Logic Network Interface Programmable Logic Memory **Figure 2. An Architecture for Adaptation**



	L1 Cache	L2 Cache
Line Size	32B or 64B	32B or 64B
Associativity	1	2
Cache Size	32KB	512KB
Write Policy	Write back + Write allocate	Write back + Write allocate
Replacement Policy	Random	Random
Transfer Rate	(L1-L2) 16B/5 cycles	(L2-Mem) 8B/15 cycles

Table 1. Simulation Parameters

	 Processor virtual addresses/data L1 Cache data physical addresses Prefetcher Prefetcher Logic additional addresses L2 Cache **Figure 4. Organizations of Prefetcher Logic**
and a data prefetch unit coupled to the first memory, wherein the data prefetch unit retrieves only computational data required by the algorithm from a second memory of second characteristic memory bandwidth and/or memory utilization and places the retrieved computational data in the first memory wherein the data prefetch unit operates independent of and in parallel with logic blocks using the computional [*sic*] data, and wherein at least the first memory and data prefetch unit are configured to conform to needs of the algorithm, and the data prefetch unit is configured to match format and location of data in the second memory.	Zhang discloses a data prefetch unit coupled to the first memory, wherein the data prefetch unit retrieves only computational data required by the algorithm from a second memory of second characteristic memory bandwidth and/or memory utilization and places the retrieved computational data in the first memory wherein the data prefetch unit operates independent of and in parallel with logic blocks using the computational data, and wherein at least the first memory and data prefetch unit are configured to conform to needs of the algorithm, and the data prefetch unit is configured to match format and location of data in the second memory. For example: **Zhang discloses a prefetcher (e.g., a data prefetch unit) implemented by programmable logic and integrated with the L1 cache (e.g., coupled to the first memory).** ("Figure 4 shows the prefetcher implementation using programmable logic integrated with the L1 cache. The prefetcher requires two pieces of application-specific information: the address ranges and the memory layout of the target data structures. The address range is needed to indicate memory bounds

Explore Litigation Insights

Docket Alarm provides insights to develop a more informed litigation strategy and the peace of mind of knowing you're on top of things.

Real-Time Litigation Alerts

Keep your litigation team up-to-date with **real-time alerts** and advanced team management tools built for the enterprise, all while greatly reducing PACER spend.

Our comprehensive service means we can handle Federal, State, and Administrative courts across the country.

Advanced Docket Research

With over 230 million records, Docket Alarm's cloud-native docket research platform finds what other services can't. Coverage includes Federal, State, plus PTAB, TTAB, ITC and NLRB decisions, all in one place.

Identify arguments that have been successful in the past with full text, pinpoint searching. Link to case law cited within any court document via Fastcase.

Analytics At Your Fingertips

Learn what happened the last time a particular judge, opposing counsel or company faced cases similar to yours.

Advanced out-of-the-box PTAB and TTAB analytics are always at your fingertips.

API

Docket Alarm offers a powerful API (application programming interface) to developers that want to integrate case filings into their apps.

LAW FIRMS
Build custom dashboards for your attorneys and clients with live data direct from the court.

Automate many repetitive legal tasks like conflict checks, document management, and marketing.

FINANCIAL INSTITUTIONS
Litigation and bankruptcy checks for companies and debtors.

E-DISCOVERY AND LEGAL VENDORS
Sync your system to PACER to automate legal marketing.