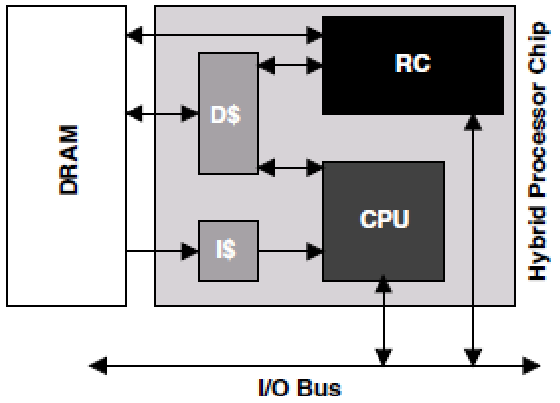
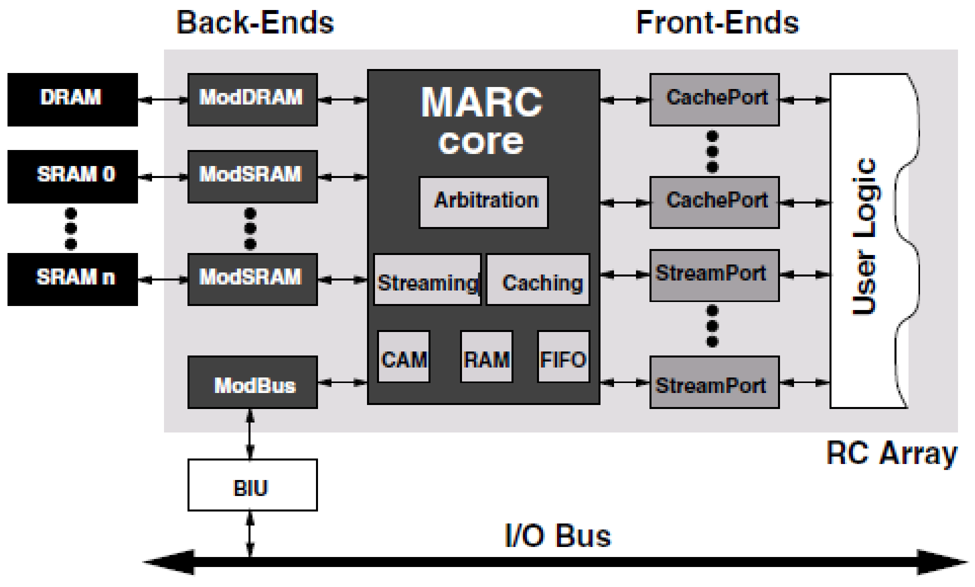
Exhibit B-3: "Memory Access Schemes for Configurable Processors," H. Lange and A. Koch, FPL 2000: Field-Programmable Logic and Applications: The Roadmap to Reconfigurable Computing, 2000 ("Lange")

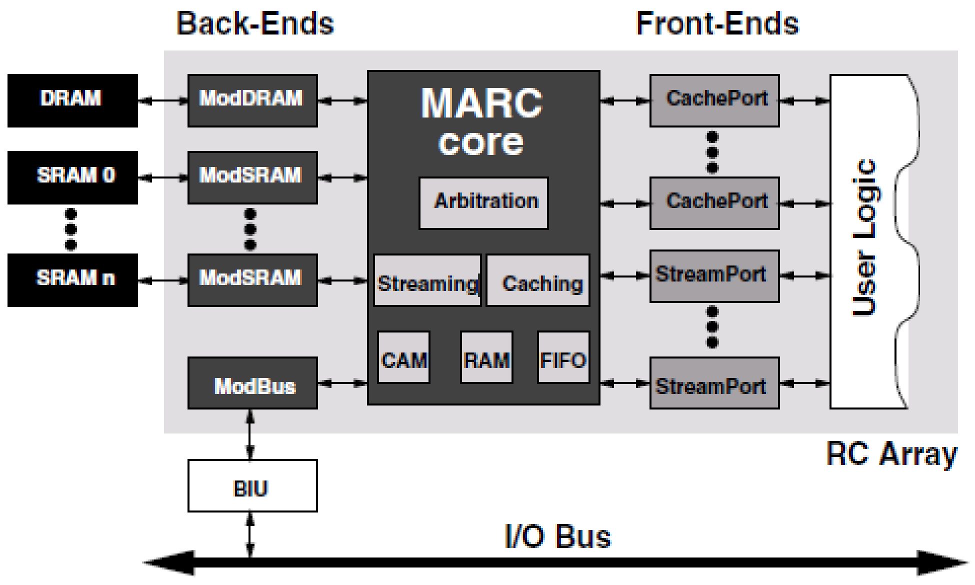
As described in the following claim chart, the asserted claims of the '867 patent are invalid in view of "Memory Access Schemes for Configurable Processors," H. Lange and A. Koch, FPL 2000: Field-Programmable Logic and Applications: The Roadmap to Reconfigurable Computing, 2000 ("Lange").

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	**Lange**
A reconfigurable processor that instantiates an algorithm as hardware comprising:	Lange discloses a reconfigurable processor that instantiates an algorithm as hardware. For example: **Lange discloses a Memory Architecture for Reconfigurable Computers (MARC) implemented in an FPGA with compute units.** Page 615 ("This work discusses the Memory Architecture for Reconfigurable Computers (MARC), a scalable, device-independent memory interface that supports both irregular (via configurable caches) and regular accesses (via pre-fetching stream buffers)."); page 616 ("Figure 1 sketches the architecture of a single-chip hybrid processor that combines fixed (CPU) and reconfigurable (RC) compute units behind a common cache (D$)."); page 624 ("We discussed one real-world implementation of MARC on an emulated hybrid processor combining a SPARC CPU with a Virtex FPGA. The sample implementation fully supports multi-threaded access to multiple memory banks as well as the creation of "virtual" memory ports attached to on-chip cache memory.")

	 DRAM D$ I$ RC CPU Hybrid Processor Chip I/O Bus **Figure 1.** Single-chip hybrid processor
a first memory having a first characteristic memory bandwidth and/or memory utilization;	Lange discloses a first memory having a first characteristic memory bandwidth and/or memory utilization. For example: **Lange discloses that the MARC core includes a cache, e.g., an L1 cache, which has a characteristic bandwidth and utilization.** Page 619 (“Using MARC, the datapath accesses memory through abstract front-end interfaces. Currently, we support two front-ends specialized for different access patterns: Caching ports provide for efficient handling of irregular accesses. Streaming ports offer a non-unit stride access to regular data structures (such as matrices or images) and perform address generation automatically. In both cases, data is pre-fetched/cached to reduce the impact of high latencies (especially for transfers using the I/O bus).”); page 621 (“In this manner, additional back-ends handling more memory banks can be attached easily. Analogously, MARC can be adapted to different FPGA technologies. For example, on the Xilinx Virtex [24] series of FPGA, the L1 cache of a caching port might be implemented using the on-chip memories.”); page 622 (“The cache is implemented as a fully associative L1 cache. It uses 4KB of Virtex Block-SelectRAM to hold the cache lines on-chip and implements write-back and random line replacement.”)

	 Back-Ends Front-Ends DRAM SRAM 0 SRAM n ModDRAM ModSRAM ModSRAM ModBus MARC core Arbitration Streaming Caching CAM RAM FIFO CachePort CachePort StreamPort StreamPort User Logic RC Array BIU I/O Bus Figure 4. MARC architecture
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	Lange 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: **Lange discloses that the MARC performs data pre-fetching of data from a memory (e.g., the second memory) through a caching port and streaming port and into the cache (i.e., the first memory).** Page 619 ("Using MARC, the datapath accesses memory through abstract front-end interfaces. Currently, we

location of data in the second memory.	support two front-ends specialized for different access patterns: Caching ports provide for efficient handling of irregular accesses. Streaming ports offer a non-unit stride access to regular data structures (such as matrices or images) and perform address generation automatically. In both cases, data is pre-fetched/cached to reduce the impact of high latencies (especially for transfers using the I/O bus).")  Back-Ends Front-Ends DRAM SRAM 0 SRAM n ModDRAM ModSRAM ModSRAM ModBus MARC core Arbitration Streaming Caching CAM RAM FIFO CachePort CachePort StreamPort StreamPort User Logic RC Array BIU I/O Bus **Figure 4.** MARC architecture **Lange discloses that MARC performs pre-fetching to address the problem of memory latency in a reconfigurable context.** Page 617 ("It is obvious from these numbers that any useful wrapper must be able to deal efficiently with access to high latency memories. This problem, colloquially known as the "memory bottleneck", has already been tackled for conventional processors using memory hierarchies (multiple cache levels) combined with techniques such as pre-fetching and streaming to improve their performance. As we will see later, these approaches are also applicable to reconfigurable systems."); page 618 ("Our goal was to learn

	from these past experiences and develop a single, scalable, and portable memory interface scheme for reconfigurable datapaths. MARC (Memory Architecture for Reconfigurable Computers) strives to be applicable for both single-chip and board-level systems, and to hide the intricacies of different memory systems from the datapath. Figure 4 shows an overview of this architecture.") **Lange discloses that the stream characteristics are matched to the specific application requirements, and that only the precise amount of data required is pre-fetched.** Page 620 ("The 'Block' register plays a crucial role in matching the stream characteristics to the specific application requirements. E.g., if the application has to process a very large string (such as in DNA matching), it makes sense for the datapath to request a large block size. The longer start-up delay (for the buffer to be filled) is amortized over the long run-time of the algorithm. For smaller amounts of data (e.g.,part of a matrix row for blocking matrix multiplication), it makes much more sense to pre-fetch only the precise amount of data required.")
'867 patent claim 3	**Lange**
The reconfigurable processor of claim 1, wherein the data prefetch unit receives processed data from on-processor memory and writes the processed data to an external off-processor memory.	Lange discloses the data prefetch unit receives processed data from on-processor memory and writes the processed data to an external off-processor memory. For example: **Lange discloses that the MARC core is located between front- and back-ends.** Page 619 ("The MARC core is located between front- and back-ends, where it acts as the main controller and data switchboard. It performs address decoding and arbitration between transfer initiators in the datapath and transfer receivers in the individual memories and busses. Logically, it can map an arbitrary number of front-ends to an arbitrary number of back-ends.")
'867 patent claim 4	**Lange**
The reconfigurable processor of claim 1, wherein the data prefetch unit comprises at	Lange discloses the data prefetch unit comprises at least one register from the

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.

WHAT WILL YOU BUILD? | sales@docketalarm.com | 1-866-77-FASTCASE

fastcase® Smarter legal research.