Exhibit 8

U.S. Patent No. 7,917,680 ("'680 Patent")

Accused Products

Apple products including smartphones, tablets, and computers with Apple A-Series or M-Series SoCs ("A System") infringe at least Claim 1 of the '680 Patent.

Claim 1

Claim 1	Apple Ax/Mx System
1[pre]. A method for managing packet-based communications in a rules-based system, the method comprising:	To the extent the preamble is limiting, the Apple Ax/Mx System im for managing packet-based communications in a rules-based systen For example, the Apple Ax/Mx System includes an ARM AMBA A similar, such as a PL301 fabric implementing ordering rules, includ limitation an "Apple Fabric." *See, e.g.*:

Claim 1	Apple Ax/Mx System
	Apple A11 Bionic: The fastest six-co processor around What's new? First, it's a six-core processor. "Exciting," you think, and indeed it is just remember that more cores on a phone isn't the same as more cores on yo laptop. There are two "performance" cores and four "high-efficiency cores" for a total o six. This chip design is similar to big.LITTLE, created by British company ARM. Without going too in-depth, this is known as 'heterogeneous computing', which effectively allows devices to run on processors that have cores of different specifications. This unlike a processor in your PC, where all the processor cores exactly the same. Compared to last year's A10 Fusion, Apple claims 25% better performance from those all-important performance cores, while the four high-efficiency cores will collectively be 70% faster for background tasks. What's more exciting is that the controller, which decides what cores get which tasks, now allows all six cores to work on the same task, allowing for 70% faster multi-threaded workloads. In terms of graphics performance, the GPU is said to be 30% faster than last ye while consuming half the power when working at the same rate as the A10. Th impressive and will help battery life when it comes to gaming. Big.LITTLE-like designs aren't unique to Apple's chip designs; the likes of the Samsung Exynos 8895 and the Qualcomm Snapdragon 835 SoC feature this s architecture. But what's new here is the doubling of the number of lower-pow http://www.trustedreviews.com/news/apple-a11-processor-specs-pe benchmarks-3286346

Claim 1	Apple Ax/Mx System

```
#define SWTCH_FAB_BASE_ADDR      (IO_BASE + 0x000800000)      // Switch Fabric Config

#define CP_COM_BASE_ADDR         (IO_BASE + 0x000D00000)      // CP Common Registers
#define CP_COM_INT_BASE_ADDR         (CP_COM_BASE_ADDR + 0x10000)     // CP Common Interrupt C

#define CP_0_DT_DBG_CA0_ADDR         (IO_BASE + 0x000D40000)      // Dup Tag Debug backdoor Ra
#define CP_1_DT_DBG_CA0_ADDR         (IO_BASE + 0x000E40000)      // Dup Tag Debug backdoor Ra

#define ACC_BASE_ADDR            (IO_BASE + 0x002000000)      // Apple Compute Complex
#define CCC_CPU0_SYS_BASE_ADDR       (IO_BASE + 0x002000000)      // Hurricane/Zephyr CPU0 Imp
#define CCC_CPU1_SYS_BASE_ADDR       (IO_BASE + 0x002100000)      // Hurricane/Zephyr CPU1 Imp

#define SOC_BUSMUX_BASE_ADDR         (IO_BASE + 0x004000000)      // AF SoC BusMux Config
```

iboot-master\platform\t8010\include\platform\soc\hwregbase_t801

```
// CPU_Fabric_Widget_Registers (base address @ CPU_FABRIC_BASE_ADDR)
#define CPU_Fabric_pl301Wrap0_AMCRDRATELIMIT        (0x0000)    // AMC Read Rate Limit R
#define CPU_Fabric_pl301Wrap0_AMCWRALIMIT       (0x0004)    // AMC Write Rate Limit Regi
#define CPU_Fabric_pl301Wrap0_AMCRTRLIMIT       (0x0008)    // AMC Read Transactions Lim
#define CPU_Fabric_pl301Wrap0_AMCWTRLIMIT       (0x000c)    // AMC Write Transactions Li
#define CPU_Fabric_pl301Wrap0_SPURDRATELIMIT        (0x0040)    // SPU Read Rate Limit R
#define CPU_Fabric_pl301Wrap0_SPUWRALIMIT       (0x0044)    // SPU Write Rate Limit Regi
#define CPU_Fabric_pl301Wrap0_SPURTRLIMIT       (0x0048)    // SPU Read Transactions Lim
#define CPU_Fabric_pl301Wrap0_SPUWTRLIMIT       (0x004c)    // SPU Write Transactions Li
#define CPU_Fabric_pl301Wrap0_SPUWGATHER        (0x0050)    // SPU Write Gather Register
#define CPU_Fabric_pl301Wrap0_LIOWGATHER        (0x0090)    // LIO Write Gather Register
#define CPU_Fabric_pl301Wrap0_AUERDRATELIMIT        (0x00c0)    // AUE Read Rate Limit R
#define CPU_Fabric_pl301Wrap0_AUEWRALIMIT       (0x00c4)    // AUE Write Rate Limit Regi
#define CPU_Fabric_pl301Wrap0_AUERTRLIMIT       (0x00c8)    // AUE Read Transactions Lim
#define CPU_Fabric_pl301Wrap0_AUEWTRLIMIT       (0x00cc)    // AUE Write Transactions Li
#define CPU_Fabric_pl301Wrap0_AUEWGATHER        (0x00d0)    // AUE Write Gather Register
#define CPU_Fabric_pl301Wrap0_ANSRDRATELIMIT        (0x0100)    // ANS Read Rate Limit R
#define CPU_Fabric_pl301Wrap0_ANSWRALIMIT       (0x0104)    // ANS Write Rate Limit Regi
#define CPU_Fabric_pl301Wrap0_ANSRTRLIMIT       (0x0108)    // ANS Read Transactions Lim
#define CPU_Fabric_pl301Wrap0_ANSWTRLIMIT       (0x010c)    // ANS Write Transactions Li
#define CPU_Fabric_pl301Wrap0_ANSWGATHER        (0x0110)    // ANS Write Gather Register
#define CPU_Fabric_pl301Wrap0_AXI0_ARCHANARBMI0     (0x0408)    // Configured AR channel

// NRT_Fabric_Widget_Registers (base address @ NRT_FABRIC_BASE_ADDR)
#define NRT_Fabric_pl301Wrap1_AMCRDRATELIMIT        (0x0000)    // AMC Read Rate Limit R
#define NRT_Fabric_pl301Wrap1_AMCWRALIMIT       (0x0004)    // AMC Write Rate Limit Regi
#define NRT_Fabric_pl301Wrap1_AMCRTRLIMIT       (0x0008)    // AMC Read Transactions Lim
#define NRT_Fabric_pl301Wrap1_AMCWTRLIMIT       (0x000c)    // AMC Write Transactions Li
#define NRT_Fabric_pl301Wrap1_MSRRDRATELIMIT        (0x0040)    // MSR Read Rate Limit R
#define NRT_Fabric_pl301Wrap1_MSRWRALIMIT       (0x0044)    // MSR Write Rate Limit Regi
#define NRT_Fabric_pl301Wrap1_MSRRTRLIMIT       (0x0048)    // MSR Read Transactions Lim
#define NRT_Fabric_pl301Wrap1_MSRWTRLIMIT       (0x004c)    // MSR Write Transactions Li
#define NRT_Fabric_pl301Wrap1_SDIORDRATELIMIT       (0x0080)    // SDIO Read Rate Limit
#define NRT_Fabric_pl301Wrap1_SDIOWRALIMIT      (0x0084)    // SDIO Write Rate Limit Reg
#define NRT_Fabric_pl301Wrap1_SDIORTRLIMIT      (0x0088)    // SDIO Read Transactions Li
#define NRT_Fabric_pl301Wrap1_SDIOWTRLIMIT      (0x008c)    // SDIO Write Transactions L
```

iboot-master\platform\t8002\include\platform\soc\miu.h

Claim 1	Apple Ax/Mx System
	A5.3 Transaction ordering A master can use the **AWID** and **ARID** transaction IDs to indicate its ordering ordering of transactions are as follows: • Transactions from different masters have no ordering restrictions. They can com • Transactions from the same master, but with different ID values, have no orderi complete in any order. • The data transfers for a sequence of read transactions with the same **ARID** value n in which the master issued the addresses, see *Read ordering*. • The data transfers for a sequence of write transactions with the same **AWID** value in which the master issued the addresses, see *Normal write ordering* and *AXI3 w* page A5-79. • There are no ordering restrictions between read and write transactions using a co **ARID**, see *Read and write interaction* on page A5-80. • *Interconnect use of transaction identifiers* on page A5-80 describes how the AX transaction ID values issued by AXI masters and slaves. AMBA AXI and ACE Protocol specification, Issue D, ARM, Oct.
1[a] in a packet processor, generating a protocol-based ordering configuration for passing packet data as a function of protocol compliance rules for the rules-based system, and	The Apple Ax/Mx System includes a packet processor that implem protocol-based ordering configuration for passing packet data as a f compliance rules for the rules-based system. For example, the Apple Ax/Mx System orders AXI packet data cor according to rules set out in the AXI3, AXI4, PL301, and/or other r *See, e.g.*:

DOCKET ALARM

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.