UNITED STATES DISTRICT COURT
FOR THE WESTERN DISTRICT OF TEXAS
WACO DIVISION

TELEPUTERS, LLC, Plaintiff v. FUJITSU AMERICA, INC., et al. Defendants	**Case No. 6:20-cv-640-ADA** **JURY TRIAL DEMANDED**

AMENDED COMPLAINT FOR PATENT INFRINGEMENT

Plaintiff Teleputers, LLC ("Plaintiff" or "Teleputers") hereby files this Original Complaint for Patent Infringement against Defendants Fujitsu America, Inc., Fujitsu Computer Products of America, Inc. (formerly known as Fujitsu Semiconductor America, Inc., which was formerly known as Fujitsu Microelectronics America, Inc.), Fujitsu Limited and Fujitsu Frontech North America, Inc. (collectively "Defendant" or "Fujitsu Entities")[1] and alleges, on information

[1] Fujitsu has a multitude of corporate structures making it difficult for the public to identify the correct Fujitsu "entities" to list as defendants in this lawsuit. *See e.g.* https://en.wikipedia.org/wiki/Fujitsu#cite_note-80. In September 2020, Four Defendant Fujitsu entities' filed and served individual motions to dismiss never indicating whether they are (or are not) the correct entities to even be in this lawsuit. Specifically, the Fujitsu entities do not deny that they make, use, sell, and/or import the Accused Instrumentalities and/or the '014 Accused Instrumentalities. *See* Dkt. Nos. 15,16,17,18.

Fujitsu could have easily avoided the expense of filing such motions by simply informing Plaintiff that they are not the correct entities to be named as a party under the multitude of existing Fujitsu shell companies. But, they refused to do so. Opting instead to file 4 (four) separate Motions to Dismiss alleging venue is improper under Rule 12(b)(3). In doing so, Fujitsu seems to have forgotten it has well over $500,000 worth of its real property located in this District. Based on all 4 (four) of the self-serving declarations filed with their motions to dismiss, it is telling that none of them deny Fujitsu has joint enterprises in this district. After all, Fujitsu has over a half million dollars worth of computer and server related assets located in someone's offices, if not their own, then obviously their joint enterprise partner(s). Those asserts and belief, as follows:

THE PARTIES

1. Teleputers, LLC is a limited liability company organized and existing under the laws of the State of New Jersey with its principal place of business in Princeton, New Jersey.

2. On information and belief, Fujitsu America, Inc. is a California corporation, having its principal place of business at 1250 East Arques Avenue, Sunnyvale, California 94085. Fujitsu America, Inc. can be served with process through its registered agent, C T Corporation System, located at 1999 Bryan St., Ste. 900, Dallas, Texas 75201. Fujitsu America, Inc. does business in the State of Texas and in the Western District of Texas.

3. On information and belief, Fujitsu Microelectronics America, Inc. is a California corporation, having its principal place of business at 1250 East Arques Avenue, Sunnyvale, CA 94085. Upon information and belief, on or around April 19, 2010, Fujitsu Microelectronics America, Inc. changed its name to Fujitsu Semiconductor America, Inc. Fujitsu Semiconductor America, Inc. can be served with process through its registered agent, C T Corporation System, located at 1999 Bryan St., Ste. 900, Dallas, Texas 75201. Fujitsu Semiconductor America, Inc. does business in the State of Texas and in the Western District of Texas.

4. On information and belief, Fujitsu Computer Products of America, Inc. is a California corporation, having its principal place of business at 1255 East Arques Avenue, Sunnyvale, California 94085. Fujitsu Computer Products of America, Inc. can be served with process through its registered agent, C T Corporation System, located at 1999 Bryan St., Ste. 900, Dallas,

are personal property, not real property. They aren't being kept out in the rain or in a field, but instead at specific locations identified in this Amended Complaint. Going forward, again, Plaintiff asks Fujitsu to simply communicate with Plaintiff, so the correct entities are before the Court to resolve the actual dispute concerning three separate Counts of patent infringement, as required under Rule 1 of the Federal Rules of Civil Procedure.

Texas 75201. Fujitsu Computer Products of America, Inc. does business in the State of Texas and in the Western District of Texas.

5. On information and belief, Fujitsu Frontech North America, Inc., is located at 2801 Network Blvd, Frisco, Texas 75034. Fujitsu Frontech North America, Inc. can be served with process through its registered agent, C T Corporation System, located at 1999 Bryan St., Ste. 900, Dallas, Texas 75201. Fujitsu Frontech North America, Inc. does business in the State of Texas and in the Western District of Texas.

6. On information and belief, Defendant Fujitsu Limited is a company organized under the laws of Japan with its principal place of business at Shiodome City Center, 1-5-2 Higashi-Shimbashi, Minato-ku, Tokyo 105-7123, Japan. Fujitsu Limited may be served through its U.S. subsidiaries, *supra*.

JURISDICTION AND VENUE

7. This action arises under the patent laws of the United States, 35 U.S.C. § 1, *et seq*. This Court has subject matter jurisdiction under 28 U.S.C. §§ 1331 and 1338(a).

8. Defendants have committed acts of infringement in this judicial district.

9. On information and belief, Defendants maintain regular and systematic business interests in this district and throughout the State of Texas including through their representatives, employees and physical facilities.

10. On information and belief, the Court has personal jurisdiction over Defendants because Defendants have committed, and continue to commit, acts of infringement in the State of Texas, have conducted business in the State of Texas, and/or have engaged in continuous and systematic activities in the State of Texas. On information and belief, Defendants' accused instrumentalities

that are alleged herein to infringe were and continue to be used, imported, offered for sale, and/or sold in the Western District of Texas.

11. On information and belief, Defendant voluntarily conducts business and solicit customers in the State of Texas and within this District.

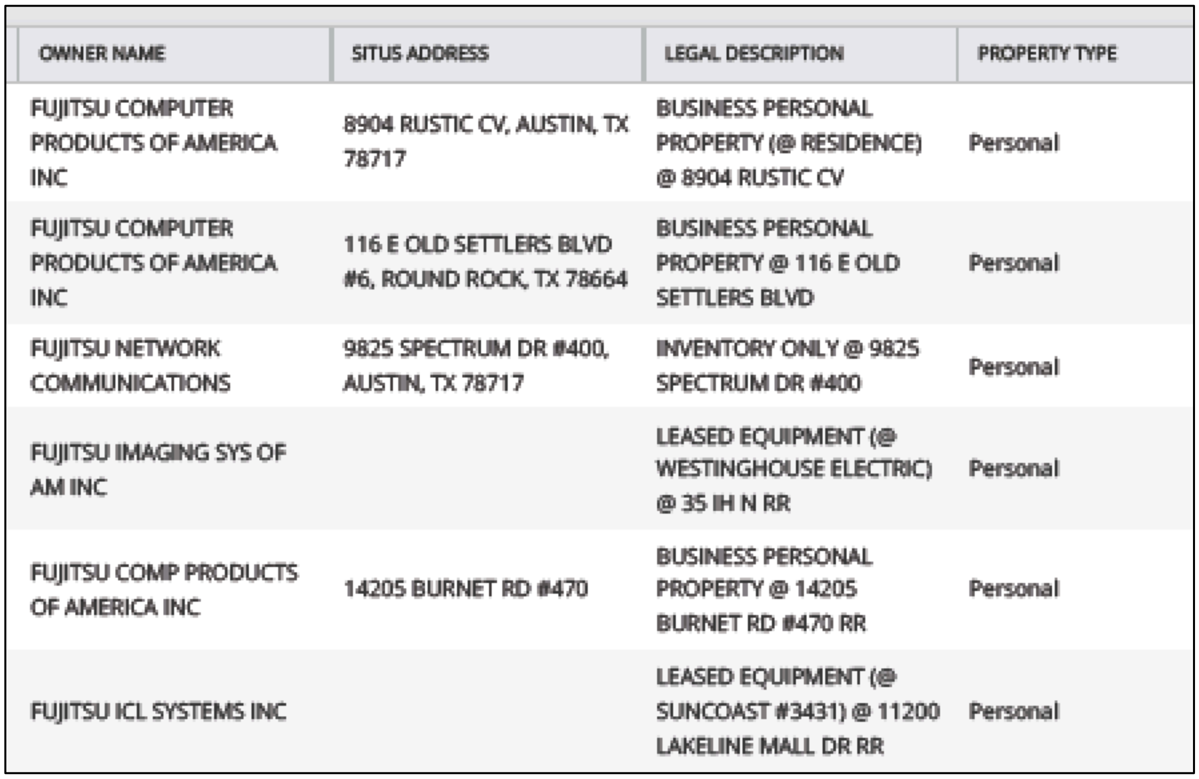
12. On information and belief, Defendant has entered into a joint enterprise with one or more technology companies in Austin leading to Defendant invest and remain owner in over $500,000 worth of its real property in this District, including, but not limited to, the following listed below.

(a) Located at 2802 Flintrock TRCE 201, Austin, TX 78734, Fujitsu has at least $46,414 worth of personal property at this location.

(b) Located at 1831 Wells Branch Pkwy, TX 78728, Defendant has at least $34,321 worth of personal property at this location.

(c) Located at 8904 Rustic CV, Austin, TX 78717, Defendant has $24508 worth of its own personal property at this location.

(d) Located at 116 E Old Settlers Blvd, #6, Round Rock, TX 87664, Defendant has $439,325 worth of its own personal property at this location.



OWNER NAME	SITUS ADDRESS	LEGAL DESCRIPTION	PROPERTY TYPE
FUJITSU COMPUTER PRODUCTS OF AMERICA INC	8904 RUSTIC CV, AUSTIN, TX 78717	BUSINESS PERSONAL PROPERTY (@ RESIDENCE) @ 8904 RUSTIC CV	Personal
FUJITSU COMPUTER PRODUCTS OF AMERICA INC	116 E OLD SETTLERS BLVD #6, ROUND ROCK, TX 78664	BUSINESS PERSONAL PROPERTY @ 116 E OLD SETTLERS BLVD	Personal
FUJITSU NETWORK COMMUNICATIONS	9825 SPECTRUM DR #400, AUSTIN, TX 78717	INVENTORY ONLY @ 9825 SPECTRUM DR #400	Personal
FUJITSU IMAGING SYS OF AM INC		LEASED EQUIPMENT (@ WESTINGHOUSE ELECTRIC) @ 35 IH N RR	Personal
FUJITSU COMP PRODUCTS OF AMERICA INC	14205 BURNET RD #470	BUSINESS PERSONAL PROPERTY @ 14205 BURNET RD #470 RR	Personal
FUJITSU ICL SYSTEMS INC		LEASED EQUIPMENT (@ SUNCOAST #3431) @ 11200 LAKELINE MALL DR RR	Personal

13. On information and belief, Defendants generate substantial revenue within this District and from the acts of infringement as carried out in this District. As such, the exercise of jurisdiction over Defendants would not offend the traditional notions of fair play and substantial justice.

14. Venue is proper in the Western District of Texas pursuant to 28 U.S.C. § 1400(b) and 28 U.S.C. § 1391(c)(3).

NOTICE OF TELEPUTERS' PATENTS

15. Teleputers is owner by assignment of U.S. Patent No. 6,922,472 ("the '472 Patent") entitled "Method and system for performing permutations using permutation instructions based on butterfly networks." A copy may be obtained at: https://patents.google.com/patent/US6922472B2/en.

16. Teleputers is owner by assignment of U.S. Patent No. 6,952,478B2 ("the '478 Patent") entitled "Method and system for performing permutations using permutation instructions based on modified omega and flip stages." A copy may be obtained at: https://patents.google.com/patent/US6952478B2/en.

17. Teleputers is owner by assignment of U.S. Patent No. 7,092,526B2 ("the '526 Patent") entitled "Method and system for performing subword permutation instructions for use in two-dimensional multimedia processing." A copy may be obtained at: https://patents.google.com/patent/US7092526B2/en.

18. Teleputers is owner by assignment of U.S. Patent No. 7,174,014B2 ("the '014 Patent" and collectively with the '478 Patent and the '526 Patent, "the Patents-in-Suit") entitled "Method and system for performing permutations with bit permutation instructions." A copy may be obtained at: https://patents.google.com/patent/US7174014B2/en.

19. Teleputers is owner by assignment of U.S. Patent No. 7,519,795B2 ("the '795 Patent") entitled "Method and system for performing permutations with bit permutation instructions." A copy may be obtained at: https://patents.google.com/patent/US7519795B2/en.

20. The foregoing Patents, namely the '014 Patent, the '526 Patent, the '478 Patent, the '472 Patent, and the '795 Patent are collectively referred to as "the Teleputers Patents."

21. Teleputers is the owner of all right, title, and interest in each of the Teleputers Patents. None of the Teleputers Patents, nor any of the claimed subject matter in any such Teleputers Patents, has been otherwise assigned to any person or entity other than Teleputers. Teleputers therefore has complete and unfettered standing to assert and seek money damages for the infringement of each and every one of the Teleputers Patents.

22. No entity other than Teleputers presently claims any ownership interest, valid or otherwise, in any of the Teleputers Patents. Teleputers possesses full legal title to each of the Teleputers Patents.

23. The records at the United States Patent and Trademark Office indicate duly recorded assignments of the Teleputers Patents from the inventors (Lee, Shi, Yang, and/or Vachharajani) to Teleputers, LLC, executed on February 14, 2005. No other assignments of interest in any Teleputers Patent have been recorded with the United States Patent and Trademark Office, and no such assignments exist. Indeed, the face of each Teleputers Patent properly identifies Teleputers LLC as the legal assignee. As such, because each of the Teleputers Patents were issued to the inventors, and because the inventors assigned the Teleputers Patents to Teleputers LLC and filed copies of such assignments with the Patent and Trademark Office, Plaintiff presumptively has proper standing to bring these causes of action. By operation of law, legal title vests in the inventors, and passes to another only by way of assignment or effective legal transfer.

24. Princeton University claims no rights to the patents-in-suit. Even if Princeton possessed any rights whatsoever in any Teleputers Patent, such rights were equitable in nature and non-exclusive to the rights of the inventors. The Verified Statement Claiming Small Entity Status (dated March 5, 2000) in the certain Provisional Patent Application Number 60/202,250 states only that certain unidentified "rights under contract or law" were, at the time, allegedly possessed by The Trustees of Princeton University. The Verified Statement further made clear that the named inventors possessed legal rights to the inventions. At best, such rights possessed by Princeton were equitable, and were in any event limited to the *inventions*, not to the issued *patents*. Further, the written policies of Princeton University relating to inventions (*see*

https://dof.princeton.edu/policies-procedure/policies/patents) expressly call for the outright assignment of inventions to the inventors or the transfer of the inventions to a patent management company. Having not transferred any of the Teleputers Patents to any patent management company, the historical actions of Princeton reflect an abandonment of equitable rights and an assignment of all rights (equitable and legal) to the inventors. Stated differently, the conduct of the parties (Princeton and the inventors) evidences an abandonment of rights on the part of Princeton, and full equitable and legal title in the inventors. Inventors have releases from Princeton.

25. The Teleputers Patents are valid, enforceable, and were duly issued in full compliance with Title 35 of the United States Code.

26. Defendant, at least by the date of this Original Complaint, is on notice of the Teleputers Patents.

27. The ’526 Patent relates generally to methods and systems for providing provides a set of permutation primitives for current and future 2-D multimedia programs which are based on decomposing images and objects into atomic units, then finding the permutations desired for the atomic units. The subword permutation instructions for these 2-D building blocks are also defined for larger subword sizes at successively higher hierarchical levels. The atomic unit can be a 2×2 matrix and four triangles contained within the 2×2 matrix. Each of the elements in the matrix can represent a subword of one or more bits. The permutations provide vertical, horizontal, diagonal, rotational, and other rearrangements of the elements in the atomic unit. *See* Abstract, ’526 Patent.

28. The claims of the ’526 Patent claim priority to at least May 7, 2001.

29. The claims of the ’526 Patent are not drawn to laws of nature, natural phenomena, or abstract ideas. Although the systems and methods claimed in the Asserted Patents are ubiquitous now (and, as a result, are widely infringed), the specific combinations of elements, as recited in the claims, was not conventional or routine at the time of the invention.

30. Further, the claims of the ’526 Patent overcome deficiencies in the prior art, including but not limited to those deficiencies embodied in subword parallelism, shift-and-rotate instructions, extract-and-deposit instructions, and mix-and-permute. The prior art was deficient in its ability to permute more than 16 elements. *See* ’526 Patent at Col. 1:15-2:23.

31. Further, the claims of the ’526 Patent contain inventive concepts which transform the underlying non-abstract aspects of the claims into patent-eligible subject matter.

32. For example, the claims of the ’526 Patent recite and are drawn to improvements in existing computational technologies, and provide for efficient subword permutation instructions that can be used for parallel execution, for example in 2-D multimedia processing. *See* ’526 Patent at Col. 2:50-53.

33. Further, the claims of the ’526 Patent recite and are drawn to improvements in existing computational technologies, and provide for single-cycle instructions, which can be used to construct any type of permutations needed in two-dimensional (2-D) multimedia processing. The instructions can be used in a programmable processor, such as a digital signal processor, video signal processors, media processors, multimedia processors, cryptographic processors and programmable System-on-a-Chips (SOCs). *See* ’526 Patent at Col. 2:56-62.

34. Further, the claims of the ’526 Patent recite and are drawn to improvements in existing computational technologies, wherein the subword permutation primitives enhance the use of subword parallelism by allowing in-place rearrangement of packed subwords across multiple

registers, reducing the need for memory accesses with potentially costly cache misses. The alphabet of permutation primitives of the invention is easy to implement and is useful for 2-D multimedia processing and for other data-parallel computations using subword parallelism. *See* '526 Patent at Col. 3:40-47.

35. The foregoing improvements and technological solutions, as captured in the claims of the '526 Patent, enable prior art systems to perform better than they previously could by implementing unconventional methodologies.

36. Further, the claims of the '526 Patent do not preempt all methods and systems for providing permutation primitives.

37. Consequently, the claims of the '526 Patent recite systems and methods resulting in improved functionality of the claimed systems and represent technological improvements to the operation of computers.

38. The '526 Patent was examined by Primary United States Patent Examiner Hosuk Song. During the examination of the '526 Patent, the United States Patent Examiner(s) searched for prior art in the following US Classifications: 708/100, 708/520, 712/1, 10, 20, 16, 24, 200, 380/28, 380/37, 42-47.

39. After conducting a search for prior art during the examination of the '526 Patent, the United States Patent Examiner(s) identified and cited the following as the most relevant prior art references found during the search: (i) US4751733A; (ii) US4845668A; (iii) US5113516A; (iv) US5423010A; and (v) US5673321A.

40. After giving full proper credit to the prior art and having conducted a thorough search for all relevant art and having fully considered the most relevant art known at the time, the United States Patent Examiner(s) allowed all of the claims of the '526 Patent to issue. In so doing, it is

presumed that the Examiner(s) used his or her knowledge of the art when examining the claims. *K/S Himpp v. Hear-Wear Techs., LLC,* 751 F.3d 1362, 1369 (Fed. Cir. 2014). It is further presumed that the Examiner has experience in the field of the invention, and that the Examiner properly acted in accordance with a person of ordinary skill. *In re Sang Su Lee,* 277 F.3d 1338, 1345 (Fed. Cir. 2002).

41. The '472 Patent relates generally to methods and systems for providing permutation instructions which can be used in software executed in a programmable processor for solving permutation problems in cryptography, multimedia and other applications. The permute instructions are based on a Benes network comprising two butterfly networks of the same size connected back-to-back. Intermediate sequences of bits are defined that an initial sequence of bits from a source register are transformed into. Each intermediate sequence of bits is used as input to a subsequent permutation instruction. Permutation instructions are determined for permitting the initial source sequence of bits into one or more intermediate sequence of bits until a desired sequence is obtained. The intermediate sequences of bits are determined by configuration bits. The permutation instructions form a permutation instruction sequence of at least one instruction. At most 21 gr/m permutation instructions are used in the permutation instruction sequence, where r is the number of k-bit subwords to be permuted, and m is the number of network stages executed in one instruction. The permutation instructions can be used to permute k-bit subwords packed into an n-bit word, where k can be 1, 2, . . . , or n bits, and k*r=n. *See* Abstract, '472 Patent.

42. The claims of the '472 Patent claim priority to at least May 5, 2000.

43. The claims of the '472 Patent are not drawn to laws of nature, natural phenomena, or abstract ideas. Although the systems and methods claimed in the Asserted Patents are ubiquitous

now (and, as a result, are widely infringed), the specific combinations of elements, as recited in the claims, was not conventional or routine at the time of the invention.

44. Further, the claims of the ’472 Patent contain inventive concepts which transform the underlying non-abstract aspects of the claims into patent-eligible subject matter.

45. Further, the claims of the ’472 Patent overcome deficiencies in the prior art, including but not limited to those relating to secure use of the Internet, symmetric key cryptography, bit-level permutations, table lookup methods, and methods requiring excessive memory requirements. *See* ’472 Patent at Col. 1:17-3:15.

46. For example, the claims of the ’472 Patent recite and are drawn to improvements in existing computational technologies, and provide significantly faster and more economical ways to perform arbitrary permutations of n bits, without any need for table storage, which can be used for encrypting large amounts of data for confidentiality or privacy. *See* ’472 Patent at Col. 3:17-21.

47. Further, the claims of the ’472 Patent recite and are drawn to improvements in existing computational technologies, and provide improved and more efficient cryptography, which provides for improved multimedia processing. *See* ’472 Patent at Col. 3:24-37.

48. Further, the claims of the ’472 Patent recite and are drawn to improvements in existing computational technologies, and provide a basis for the design of new processors or coprocessors which can be efficient for both cryptography and multimedia software. *See* ’472 Patent at Col. 3:42-47.

49. The foregoing improvements and technological solutions, as captured in the claims of the ‘472 Patent, enable prior art systems to perform better than they previously could by implementing unconventional methodologies.

50. Further, the claims of the ’472 Patent do not preempt all methods and systems for solving permutation problems in cryptography.

51. Consequently, the claims of the ’472 Patent recite systems and methods resulting in improved functionality of the claimed systems and represent technological improvements to the operation of computers.

52. The ’472 Patent was examined by Primary United States Patent Examiner Gilberto Barron, Jr, with Assistant Examiner Grigory Gurshman. During the examination of the ’472 Patent, the United States Patent Examiner(s) searched for prior art in the following US Classifications: 380/37, 28, 1.

53. After conducting a search for prior art during the examination of the ’472 Patent, the United States Patent Examiner(s) identified and cited the following as the most relevant prior art references found during the search: (i) US5495476A; (ii) US5546393A; (iii) US6381690B1; (iv) US6446198B1; (v) US6629115B1; (vi) US6108311A; and (vii) US5940389A.

54. After giving full proper credit to the prior art and having conducted a thorough search for all relevant art and having fully considered the most relevant art known at the time, the United States Patent Examiner(s) allowed all of the claims of the ’472 Patent to issue. In so doing, it is presumed that the Examiner(s) used his or her knowledge of the art when examining the claims. *K/S Himpp v. Hear-Wear Techs., LLC,* 751 F.3d 1362, 1369 (Fed. Cir. 2014). It is further presumed that the Examiner has experience in the field of the invention, and that the Examiner properly acted in accordance with a person of ordinary skill. *In re Sang Su Lee,* 277 F.3d 1338, 1345 (Fed. Cir. 2002).

55. The ’014 Patent relates generally to methods and systems for providing permutation instructions usable in a programmable processor for solving permutation problems in

cryptography, multimedia and other applications. PPERM and PPERM3R instructions are defined to perform permutations by a sequence of instructions with each sequence specifying the position in the source for each bit in the destination. In the PPERM instruction bits in the destination register that change are updated and bits in the destination register that do not change are set to zero. In the PPERM3R instruction bits in the destination register that change are updated and bits in the destination register that do not change are copied from intermediate result of previous PPERM3R instructions. Both PPERM and PPERM3R instructions can individually do permutation with bit repetition. Both PPERM and PPERM3R instructions can individually do permutation of bits stored in more than one register. In an alternate embodiment, a GRP instruction is defined to perform permutations. *See* Abstract, '014 Patent.

56. The claims of the '014 Patent claim priority to at least May 7, 2001.

57. The claims of the '014 Patent are not drawn to laws of nature, natural phenomena, or abstract ideas. Although the systems and methods claimed in the Asserted Patents are ubiquitous now (and, as a result, are widely infringed), the specific combinations of elements, as recited in the claims, was not conventional or routine at the time of the invention.

58. Further, the claims of the '014 Patent contain inventive concepts which transform the underlying non-abstract aspects of the claims into patent-eligible subject matter.

59. Further, the claims of the '014 Patent overcome deficiencies in the prior art, including but not limited to those relating to secure use of the Internet, symmetric key cryptography, bit-level permutations, table lookup methods, and methods requiring excessive memory requirements. *See* '014 Patent at 1:14-2:67.

60. For example, the claims of the '014 Patent recite and are drawn to improvements in existing computational technologies, and provide significantly faster and more economical ways

to perform arbitrary permutations of n bits, without any need for table storage, which can be used for encrypting large amounts of data for confidentiality or privacy. *See* '014 Patent at Col.3:1-5.

61. Further, the claims of the '014 Patent recite and are drawn to improvements in existing computational technologies, and provide improved and more efficient cryptography, which provides for improved multimedia processing. *See* '014 Patent at Col. 3:9-21.

62. Further, the claims of the '014 Patent recite and are drawn to improvements in existing computational technologies, and provide a basis for the design of new processors or coprocessors which can be efficient for both cryptography and multimedia software. *See* '014 Patent at Col. 3:26-31.

63. The foregoing improvements and technological solutions, as captured in the claims of the '014 Patent, enable prior art systems to perform better than they previously could by implementing unconventional methodologies.

64. Further, the claims of the '014 Patent do not preempt all methods and systems for solving permutation problems in cryptography.

65. Consequently, the claims of the '014 Patent recite systems and methods resulting in improved functionality of the claimed systems and represent technological improvements to the operation of computers.

66. The '014 Patent was examined by Primary United States Patent Examiner Emmanuel L. Moise, with Assistant Examiner Paul Callahan. During the examination of the '014 Patent, the United States Patent Examiner(s) searched for prior art in the following US Classifications: 380/44, 380/265, 28, 377/54, 60, 75, 67, 81, 711/109, 340/825.68, 365/73, 78, 712/1, 24, 10, 712/223.

67. After conducting a search for prior art during the examination of the ’014 Patent, the United States Patent Examiner(s) identified and cited the following as the most relevant prior art references found during the search: (i) US5524256A; (ii) US5734721A; (iii) US6865272B2; (iv) US4907233A; (v) JP2863597B2; (vi) US5734334A; (vii) US5422705A; (viii) GB9617553D0; (ix) US5996104A; (x) US6275965B1; (xi) US6233671B1; (xii) US6483543B1; (xiii) EP0992916A1; (xiv) US7174014B2; and (xv) Zhijie Shi, Ruby B. Lee: “Bit Permutation Instructions for Accelerating Software Cryptography”, in Proc. IEEE Intl. Conf. Application-Specific Systems, Architectures and Processors, pp. 138-148, Jul. 2000.

68. After giving full proper credit to the prior art and having conducted a thorough search for all relevant art and having fully considered the most relevant art known at the time, the United States Patent Examiner(s) allowed all of the claims of the ’014 Patent to issue. In so doing, it is presumed that the Examiner(s) used his or her knowledge of the art when examining the claims. *K/S Himpp v. Hear-Wear Techs., LLC,* 751 F.3d 1362, 1369 (Fed. Cir. 2014). It is further presumed that the Examiner has experience in the field of the invention, and that the Examiner properly acted in accordance with a person of ordinary skill. *In re Sang Su Lee,* 277 F.3d 1338, 1345 (Fed. Cir. 2002).

ACCUSED INSTRUMENTALITIES

69. On information and belief, Defendants make, use, import, sell, and/or offer for sale a multitude of products and services as systems on chips (“SoC”) that employ Arm Neon technology supporting the infringing instructions including, but not limited to: MB86R1x, MB86R2x, MB86HD6x, MB86R24, (“Triton-C” Graphics Display Controller) (individually and collectively, the “Accused Instrumentalities”). On information and belief, the Accused

Instrumentalities are made, used, sold, offered for sale, and/or imported in the United States by Defendants.

70. On information and belief, Defendants also make, use, import, sell, and/or offer for sale a multitude of products and services and provides, for example, the SPARC64 XII processor used in Fujitsu servers (including but not limited to SPARC M12-1, SPARC M12-2 and SPARC M12-2S) (the "'014 Accused Instrumentalities"). On information and belief, the '014 Accused Instrumentalities are made, used, sold, offered for sale, and/or imported in the United States by Defendants.

COUNT I
(Infringement of U.S. Patent No. 7,092,526B2)

71. Teleputers incorporates the above paragraphs by reference.

72. Defendant has been on notice of the '526 Patent at least as early as the date it received service of this Original Complaint.

73. On information and belief, Defendant has directly infringed and continue to infringe the '526 Patent by making, using, importing, selling, and/or, offering for sale the Accused Instrumentalities in the United States.

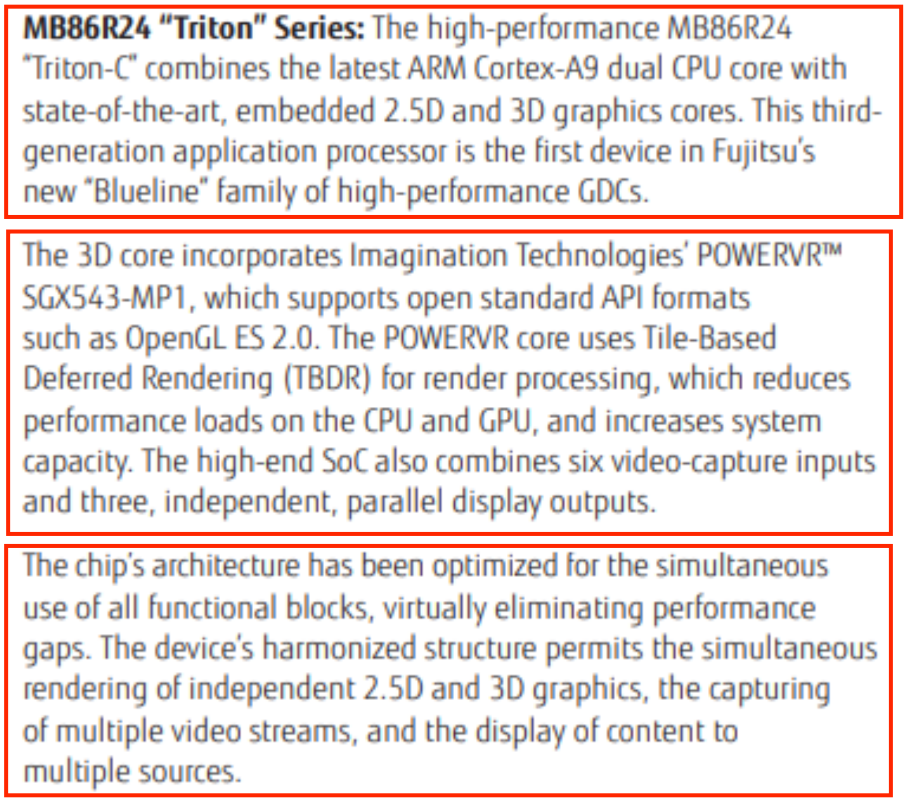
74. On information and belief, Defendant, with knowledge of the '526 Patent, indirectly infringes the '526 Patent by inducing others to infringe the '526 Patent. In particular, Defendant intends to induce customers to infringe the '526 Patent by encouraging customers to use the Accused Instrumentalities in a manner that results in infringement.

75. On information and belief, Defendant also induces others, including its customers, to infringe the '526 Patent by providing technical support for the use of the Accused Instrumentalities.

76. On information and belief, at all times Defendant owns and controls the operation of the Accused Instrumentalities in accordance with an end user license agreement.

77. On information and belief, the Accused Instrumentalities necessarily infringe one or more claims of the ‘478 Patent when used as intended.

78. On information and belief, the Accused Instrumentalities infringe at least Claim 1 of the ’526 Patent by providing a method for permuting data based on decomposing images and objects into atomic elements. For example, Defendant provides system-on-chip solutions for data processing. Defendant’s MB86R24 SoC (used herein as an exemplary product) utilizes ARM Neon technology for improving video encoding and decoding. ARM Neon SIMD architecture provides permutation instructions to rearrange individual elements present in 2D/3D graphics. Further, upon information and belief, Defendant directly infringes the claim at least when it tests its SoCs. During such tests, Defendant utilizes the SoCs to perform permutation on the input data using permutation instructions available in ARM Neon SIMD ISA (Instruction Set Architecture).

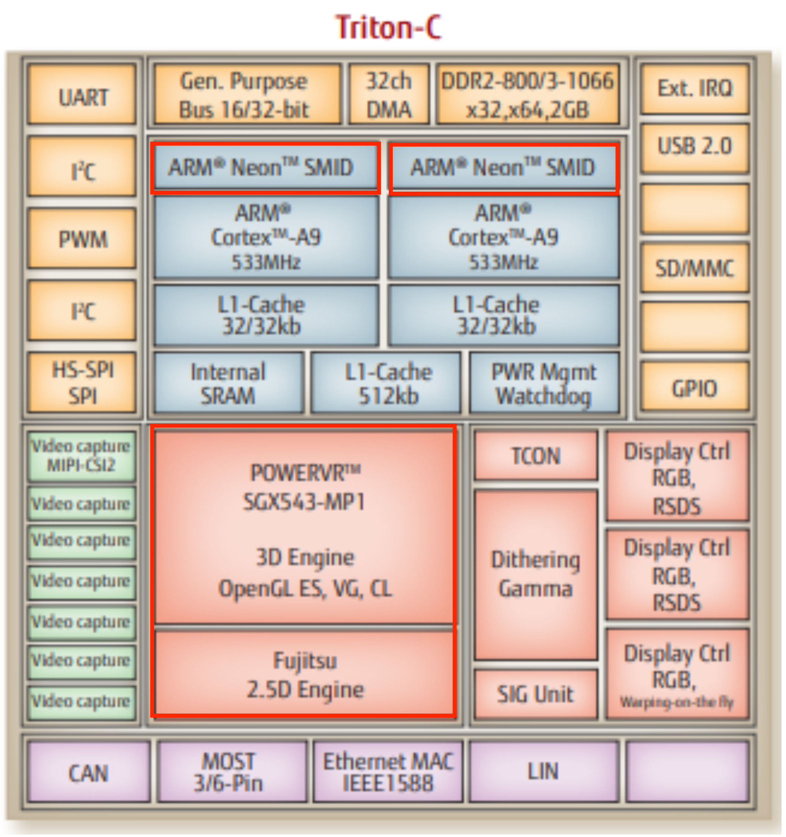


MB86R24 "Triton" Series: The high-performance MB86R24 "Triton-C" combines the latest ARM Cortex-A9 dual CPU core with state-of-the-art, embedded 2.5D and 3D graphics cores. This third-generation application processor is the first device in Fujitsu's new "Blueline" family of high-performance GDCs.

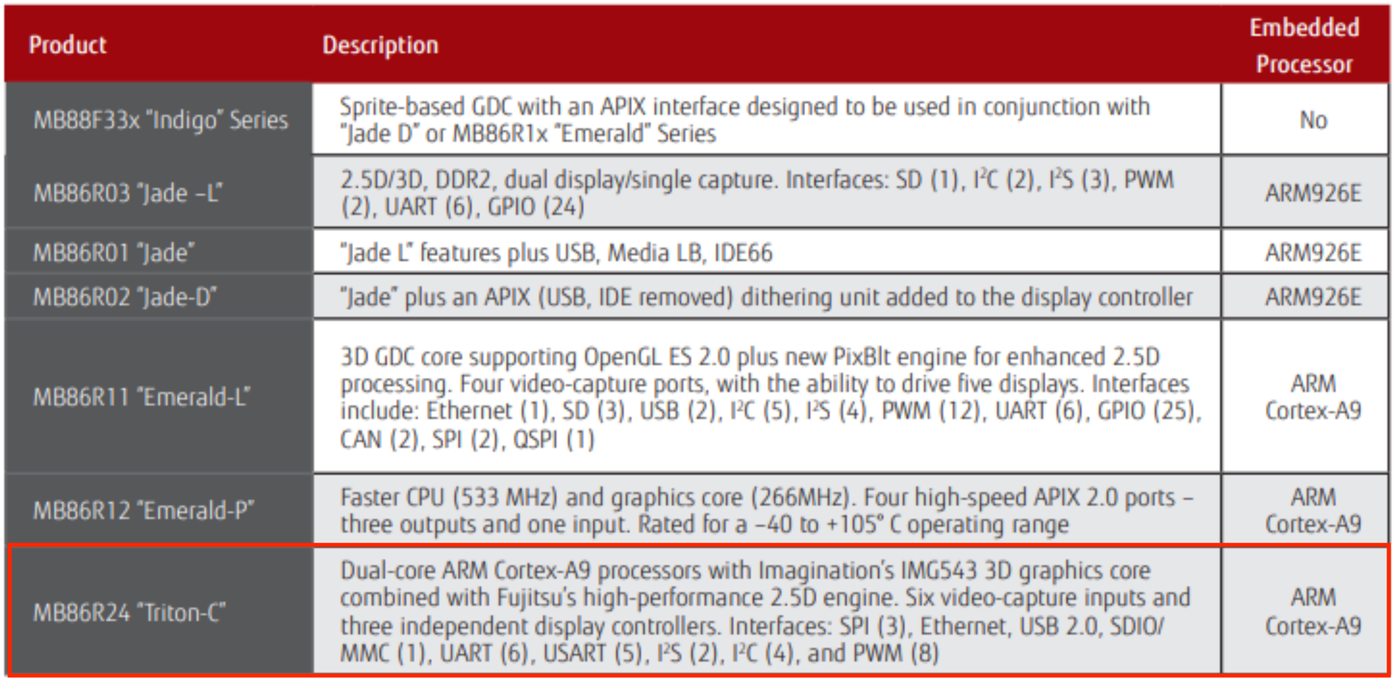
The 3D core incorporates Imagination Technologies' POWERVR™ SGX543-MP1, which supports open standard API formats such as OpenGL ES 2.0. The POWERVR core uses Tile-Based Deferred Rendering (TBDR) for render processing, which reduces performance loads on the CPU and GPU, and increases system capacity. The high-end SoC also combines six video-capture inputs and three, independent, parallel display outputs.

The chip's architecture has been optimized for the simultaneous use of all functional blocks, virtually eliminating performance gaps. The device's harmonized structure permits the simultaneous rendering of independent 2.5D and 3D graphics, the capturing of multiple video streams, and the display of content to multiple sources.

Source:https://www.fujitsu.com/us/Images/SPBG_GDC_Overview_PB.pdf, page 3

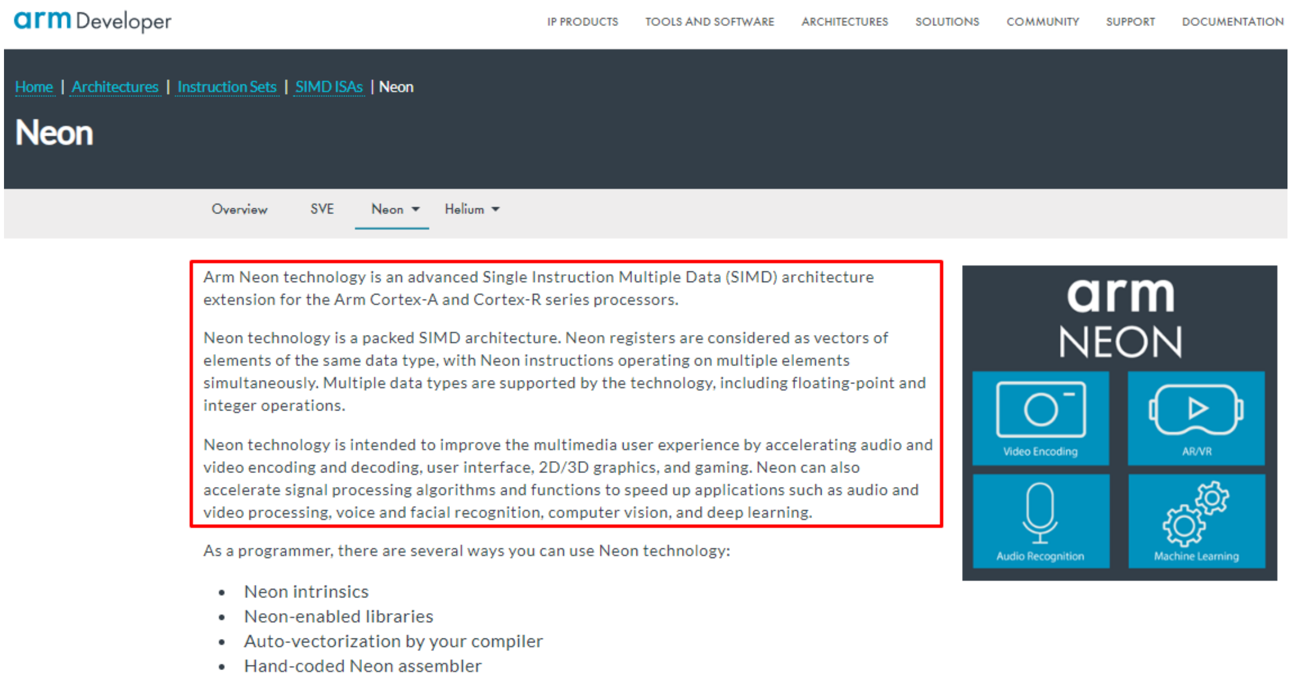


Source:https://www.fujitsu.com/us/Images/SPBG_GDC_Overview_PB.pdf, page 3



Product	Description	Embedded Processor
MB88F33x "Indigo" Series	Sprite-based GDC with an APIX interface designed to be used in conjunction with "Jade D" or MB86R1x "Emerald" Series	No
MB86R03 "Jade -L"	2.5D/3D, DDR2, dual display/single capture. Interfaces: SD (1), I²C (2), I²S (3), PWM (2), UART (6), GPIO (24)	ARM926E
MB86R01 "Jade"	"Jade L" features plus USB, Media LB, IDE66	ARM926E
MB86R02 "Jade-D"	"Jade" plus an APIX (USB, IDE removed) dithering unit added to the display controller	ARM926E
MB86R11 "Emerald-L"	3D GDC core supporting OpenGL ES 2.0 plus new PixBlt engine for enhanced 2.5D processing. Four video-capture ports, with the ability to drive five displays. Interfaces include: Ethernet (1), SD (3), USB (2), I²C (5), I²S (4), PWM (12), UART (6), GPIO (25), CAN (2), SPI (2), QSPI (1)	ARM Cortex-A9
MB86R12 "Emerald-P"	Faster CPU (533 MHz) and graphics core (266MHz). Four high-speed APIX 2.0 ports – three outputs and one input. Rated for a –40 to +105° C operating range	ARM Cortex-A9
MB86R24 "Triton-C"	Dual-core ARM Cortex-A9 processors with Imagination's IMG543 3D graphics core combined with Fujitsu's high-performance 2.5D engine. Six video-capture inputs and three independent display controllers. Interfaces: SPI (3), Ethernet, USB 2.0, SDIO/ MMC (1), UART (6), USART (5), I²S (2), I²C (4), and PWM (8)	ARM Cortex-A9

Source:https://www.fujitsu.com/us/Images/SPBG_GDC_Overview_PB.pdf, page 4



Neon

Arm Neon technology is an advanced Single Instruction Multiple Data (SIMD) architecture extension for the Arm Cortex-A and Cortex-R series processors.

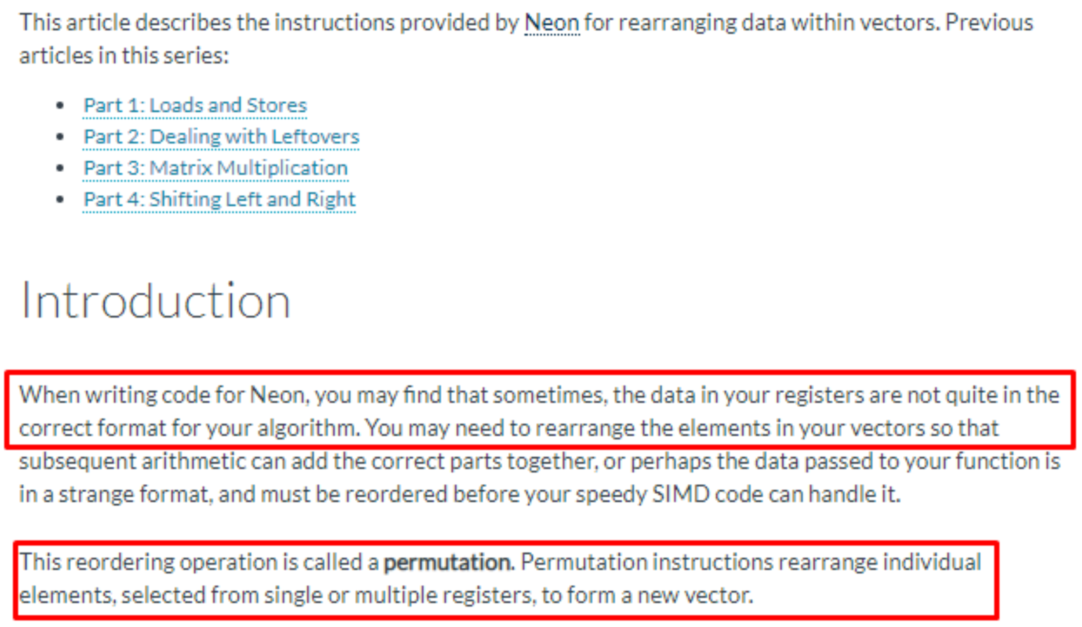
Neon technology is a packed SIMD architecture. Neon registers are considered as vectors of elements of the same data type, with Neon instructions operating on multiple elements simultaneously. Multiple data types are supported by the technology, including floating-point and integer operations.

Neon technology is intended to improve the multimedia user experience by accelerating audio and video encoding and decoding, user interface, 2D/3D graphics, and gaming. Neon can also accelerate signal processing algorithms and functions to speed up applications such as audio and video processing, voice and facial recognition, computer vision, and deep learning.

As a programmer, there are several ways you can use Neon technology:

- Neon intrinsics
- Neon-enabled libraries
- Auto-vectorization by your compiler
- Hand-coded Neon assembler

Source: https://developer.arm.com/architectures/instruction-sets/simd-isas/neon



This article describes the instructions provided by Neon for rearranging data within vectors. Previous articles in this series:

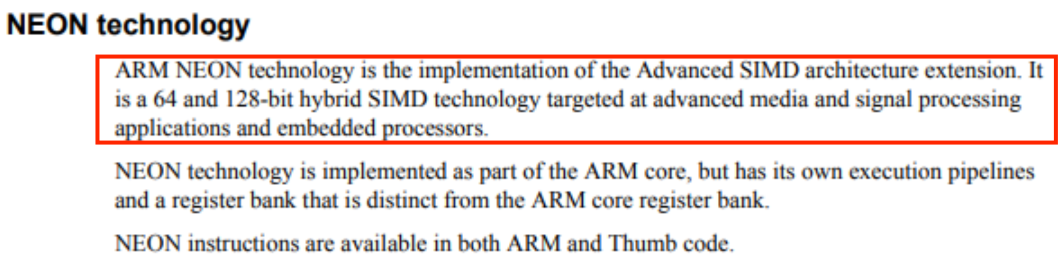
- Part 1: Loads and Stores
- Part 2: Dealing with Leftovers
- Part 3: Matrix Multiplication
- Part 4: Shifting Left and Right

Introduction

When writing code for Neon, you may find that sometimes, the data in your registers are not quite in the correct format for your algorithm. You may need to rearrange the elements in your vectors so that subsequent arithmetic can add the correct parts together, or perhaps the data passed to your function is in a strange format, and must be reordered before your speedy SIMD code can handle it.

This reordering operation is called a **permutation.** Permutation instructions rearrange individual elements, selected from single or multiple registers, to form a new vector.

Source: https://community.arm.com/developer/ip-products/processors/b/processors-ip-blog/posts/coding-for-neon---part-5-rearranging-vectors


NEON technology

ARM NEON technology is the implementation of the Advanced SIMD architecture extension. It is a 64 and 128-bit hybrid SIMD technology targeted at advanced media and signal processing applications and embedded processors.

NEON technology is implemented as part of the ARM core, but has its own execution pipelines and a register bank that is distinct from the ARM core register bank.

NEON instructions are available in both ARM and Thumb code.

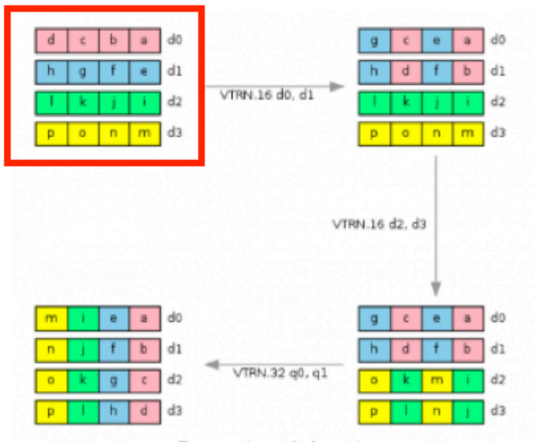
Source:

http://infocenter.arm.com/help/topic/com.arm.doc.dui0473j/DUI0473J_armasm_user_guide.pdf,

page 40

79. Further, Defendant performs and induces others to perform the step of decomposing said two dimensional data into at least one atomic element said two dimensional data being located in at least one source register said at least one atomic element of said two dimensional data is a 2×2 matrix and said two dimensional data is decomposed into data elements in said matrix. For example, the MB86R24 Triton-C uses a permutation instruction (such as a VTRN instruction) and decomposes two dimensional data (in the form of 4x4 matrix) into a 2x2 matrix. The 4x4 matrix consists of 16-bit elements ("atomic element"). The permutation instruction transposes 8, 16 or 32-bit elements between a pair of vectors. The permutation instruction is applied on the elements of the vectors by dividing it into 2x2 matrices. The two dimensional data is stored in at least one of the d0 and d1 vectors ("source registers").

Use multiple VTRN instructions to transpose larger matrices. For example, a 4x4 matrix consisting of 16-bit elements can be transposed using three VTRN instructions.



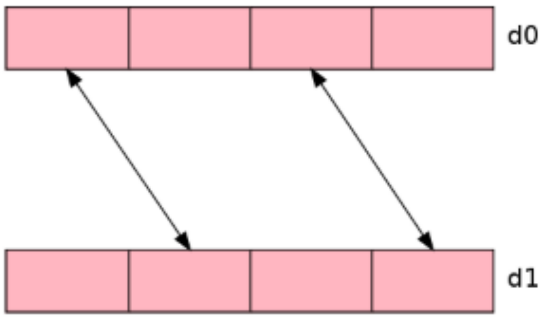
Transposing a 4x4 matrix

This is the same operation performed by VLD4 and VST4 after loading, or before storing, vectors. As they require fewer instructions, try to use these structured memory access features in preference to a sequence of VTRN instructions, where possible.

Source: https://community.arm.com/developer/ip-products/processors/b/processors-ip-blog/posts/coding-for-neon---part-5-rearranging-vectors

VTRN: Transpose

VTRN transposes 8, 16 or 32-bit elements between a pair of vectors. It treats the elements of the vectors as 2x2 matrices, and transposes each matrix.



VTRN.16 d0, d1

Source: https://community.arm.com/developer/ip-products/processors/b/processors-ip-blog/posts/coding-for-neon---part-5-rearranging-vectors

14.131 VTRN

Vector Transpose.

Syntax

`VTRN{cond}.size Qd, Qm`

`VTRN{cond}.size Dd, Dm`

where:

cond
is an optional condition code.

size
must be one of 8, 16, or 32.

Qd, Qm
specifies the vectors, for a quadword operation.

Dd, Dm
specifies the vectors, for a doubleword operation.

Operation

VTRN treats the elements of its operand vectors as elements of 2 x 2 matrices, and transposes the matrices. The following figures show examples of the operation of VTRN:

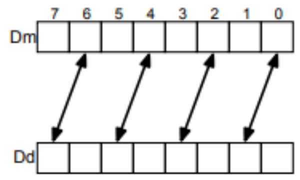


Figure 14-9 Operation of doubleword VTRN.8

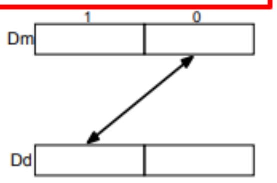


Figure 14-10 Operation of doubleword VTRN.32

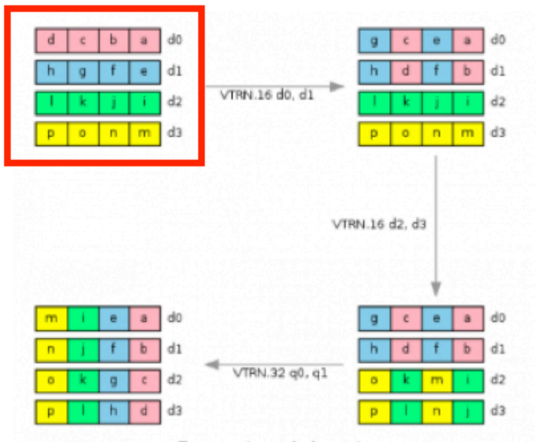
Source:

http://infocenter.arm.com/help/topic/com.arm.doc.dui0473j/DUI0473J_armasm_user_guide.pdf,

page 734.

80. Further, Defendant performs and induces others to perform the step of determining at least one permutation instruction for rearrangement of said data in said atomic element. For example, the MB86R24 Triton-C uses a permutation instruction (such as a VTRN instruction) and decomposes two dimensional data (in the form of 4x4 matrix) into a 2x2 matrix. The permutation instruction transposes ("rearrangement") 8, 16 or 32-bit elements between a pair of vectors. The permutation instruction is applied on the elements of the vectors by dividing it into 2x2 matrices.

Use multiple VTRN instructions to transpose larger matrices. For example, a 4x4 matrix consisting of 16-bit elements can be transposed using three VTRN instructions.



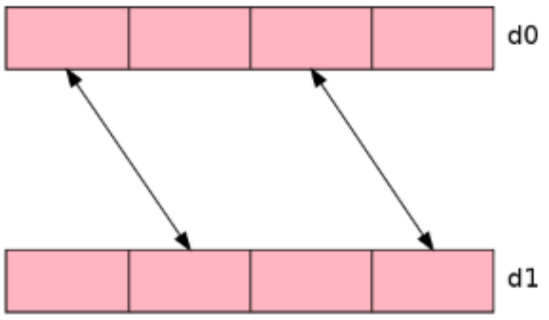
Transposing a 4x4 matrix

This is the same operation performed by VLD4 and VST4 after loading, or before storing, vectors. As they require fewer instructions, try to use these structured memory access features in preference to a sequence of VTRN instructions, where possible.

Source: https://community.arm.com/developer/ip-products/processors/b/processors-ip-blog/posts/coding-for-neon---part-5-rearranging-vectors

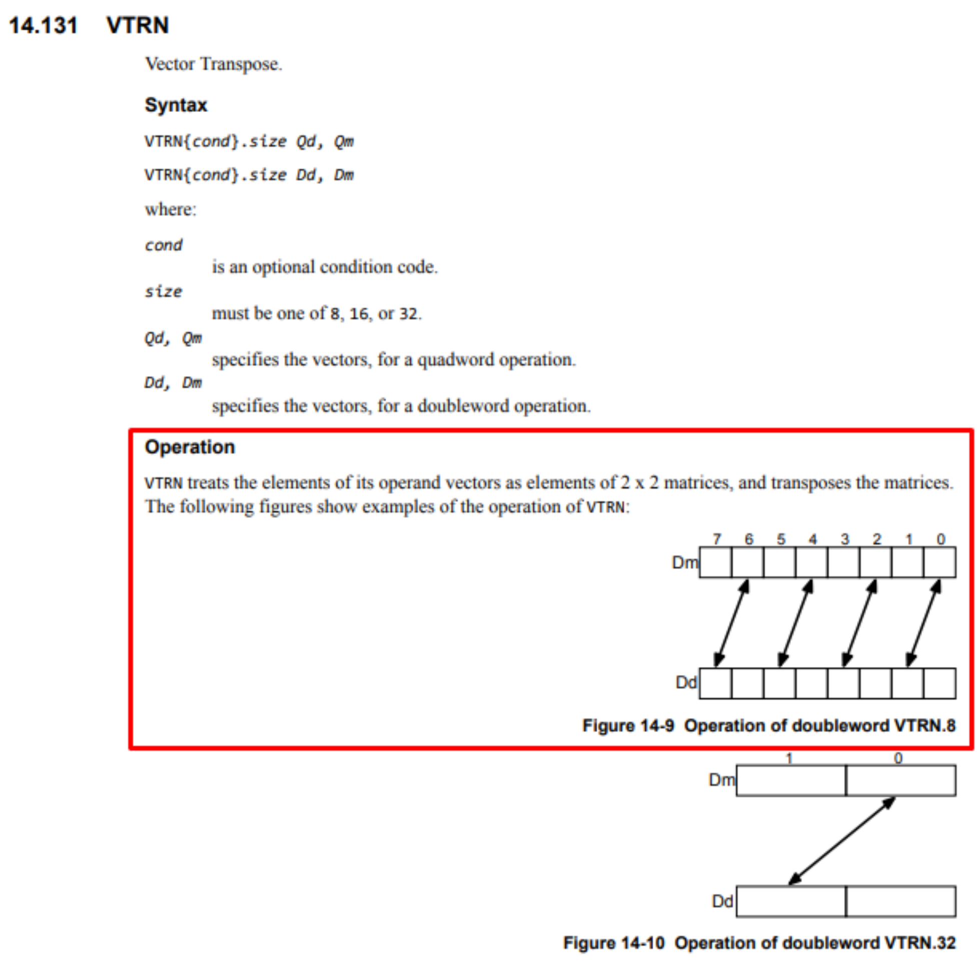
VTRN: Transpose

VTRN transposes 8, 16 or 32-bit elements between a pair of vectors. It treats the elements of the vectors as 2x2 matrices, and transposes each matrix.



VTRN.16 d0, d1

Source: https://community.arm.com/developer/ip-products/processors/b/processors-ip-blog/posts/coding-for-neon---part-5-rearranging-vectors



14.131 VTRN

Vector Transpose.

Syntax

`VTRN{cond}.size Qd, Qm`

`VTRN{cond}.size Dd, Dm`

where:

cond
is an optional condition code.

size
must be one of 8, 16, or 32.

Qd, Qm
specifies the vectors, for a quadword operation.

Dd, Dm
specifies the vectors, for a doubleword operation.

Operation

VTRN treats the elements of its operand vectors as elements of 2 x 2 matrices, and transposes the matrices. The following figures show examples of the operation of VTRN:

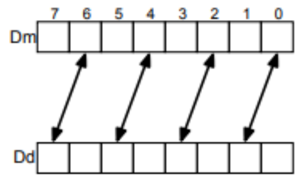


Figure 14-9 Operation of doubleword VTRN.8

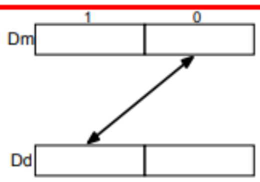
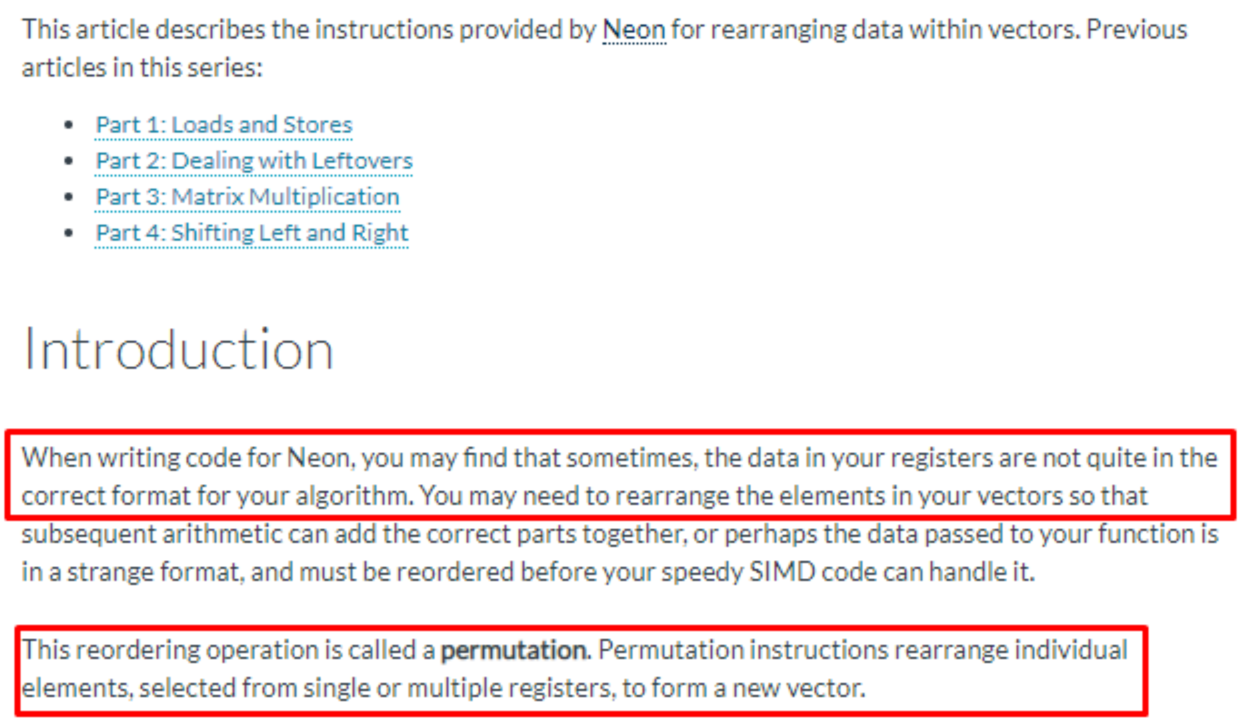


Figure 14-10 Operation of doubleword VTRN.32

Source:

http://infocenter.arm.com/help/topic/com.arm.doc.dui0473j/DUI0473J_armasm_user_guide.pdf, page 734.

81. Further, Defendant performs and induces others to perform the step of said data elements being rearranged by said at least one permutation instruction, each of said data elements representing a subword having one or more bits. For example, the MB86R24 Triton-C uses a permutation instruction (such as a VTRN instruction) and transposes ("rearrange") 8, 16 or 32-bit elements ("subwords") of the 2x2 matrix. The permutation instruction performs bit permutation on each element.



This article describes the instructions provided by Neon for rearranging data within vectors. Previous articles in this series:

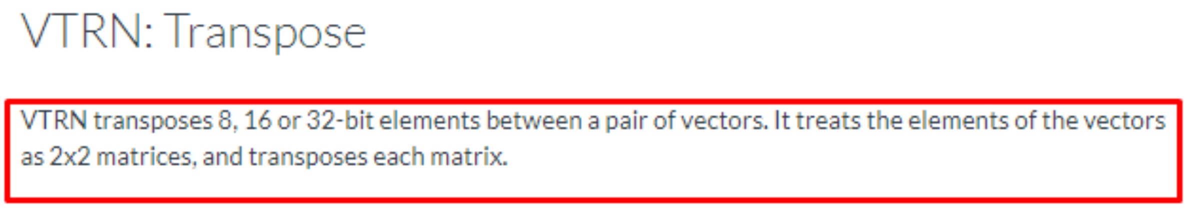
- Part 1: Loads and Stores
- Part 2: Dealing with Leftovers
- Part 3: Matrix Multiplication
- Part 4: Shifting Left and Right

Introduction

When writing code for Neon, you may find that sometimes, the data in your registers are not quite in the correct format for your algorithm. You may need to rearrange the elements in your vectors so that subsequent arithmetic can add the correct parts together, or perhaps the data passed to your function is in a strange format, and must be reordered before your speedy SIMD code can handle it.

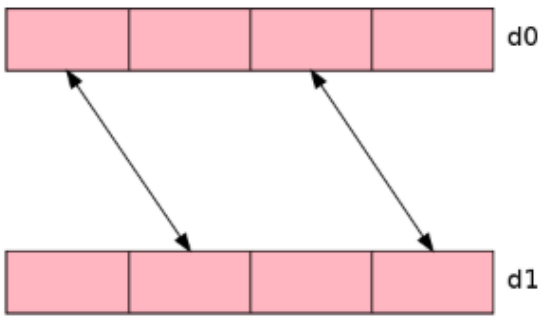
This reordering operation is called a **permutation.** Permutation instructions rearrange individual elements, selected from single or multiple registers, to form a new vector.

Source: https://community.arm.com/developer/ip-products/processors/b/processors-ip-blog/posts/coding-for-neon---part-5-rearranging-vectors



VTRN: Transpose

VTRN transposes 8, 16 or 32-bit elements between a pair of vectors. It treats the elements of the vectors as 2x2 matrices, and transposes each matrix.



VTRN.16 d0, d1

Source: https://community.arm.com/developer/ip-products/processors/b/processors-ip-blog/posts/coding-for-neon---part-5-rearranging-vectors

Operation

VTRN treats the elements of its operand vectors as elements of 2 x 2 matrices, and transposes the matrices. The following figures show examples of the operation of VTRN:

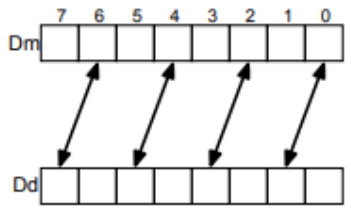


Figure 14-9 Operation of doubleword VTRN.8

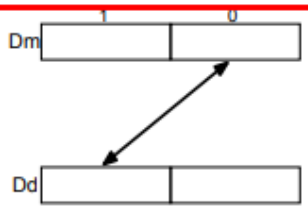
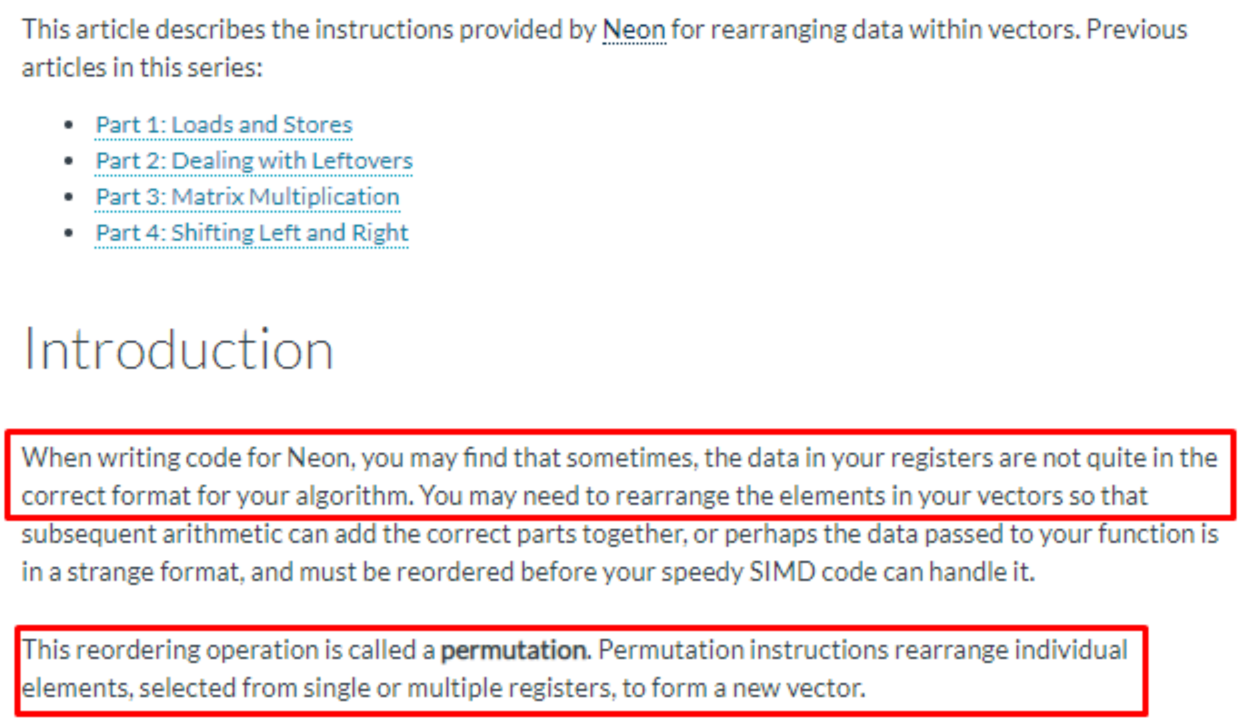


Figure 14-10 Operation of doubleword VTRN.32

Source:

http://infocenter.arm.com/help/topic/com.arm.doc.dui0473j/DUI0473J_armasm_user_guide.pdf, page 734

82. Further, Defendant performs and induces others to perform the step of applying said permutation instructions to said subwords and placing said permutated subwords into a destination register. For example, the MB86R24 Triton-C uses a permutation instruction (such as a VTRN instruction) and transposes 2x2 matrix elements to form a new vector ("placing said permutated subword into a destination register").


This article describes the instructions provided by Neon for rearranging data within vectors. Previous articles in this series:

- Part 1: Loads and Stores
- Part 2: Dealing with Leftovers
- Part 3: Matrix Multiplication
- Part 4: Shifting Left and Right

Introduction

When writing code for Neon, you may find that sometimes, the data in your registers are not quite in the correct format for your algorithm. You may need to rearrange the elements in your vectors so that subsequent arithmetic can add the correct parts together, or perhaps the data passed to your function is in a strange format, and must be reordered before your speedy SIMD code can handle it.

This reordering operation is called a **permutation**. Permutation instructions rearrange individual elements, selected from single or multiple registers, to form a new vector.

Source: https://community.arm.com/developer/ip-products/processors/b/processors-ip-blog/posts/coding-for-neon---part-5-rearranging-vectors

83. Defendant indirectly infringe the claim at least when Defendant's customers (such as device manufacturers which use Defendant's SoCs in their products) perform the method while testing their devices and when the devices are operated by end-users.

COUNT II
(Infringement of U.S. Patent No. 6,952,478B2)

84. Teleputers incorporates the above paragraphs by reference.

85. Defendant has been on notice of the '478 Patent at least as early as the date it received service of this Original Complaint.

86. On information and belief, Defendant has infringed and continue to infringe the '478 Patent by making, using, importing, selling, and/or, offering for sale the Accused Instrumentalities in the United States.

87. On information and belief, Defendant, with knowledge of the ’478 Patent, indirectly infringes the ’478 Patent by inducing others to infringe the ’478 Patent. In particular, Defendant intends to induce customers to infringe the ’478 Patent by encouraging customers to use the Accused Instrumentalities in a manner that results in infringement.

88. On information and belief, Defendant also induces others, including customers, to infringe the ’478 Patent by providing technical support for the use of the Accused Instrumentalities.

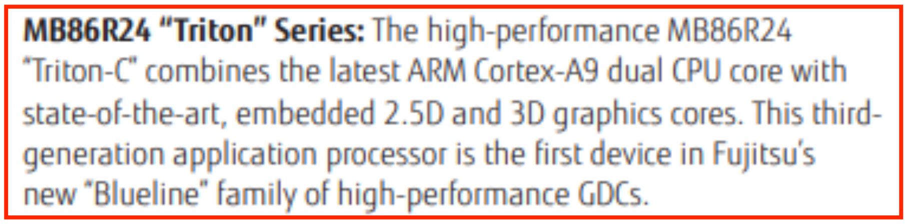
89. On information and belief, at all times Defendant owns and controls the operation of the Accused Instrumentalities in accordance with an end user license agreement.

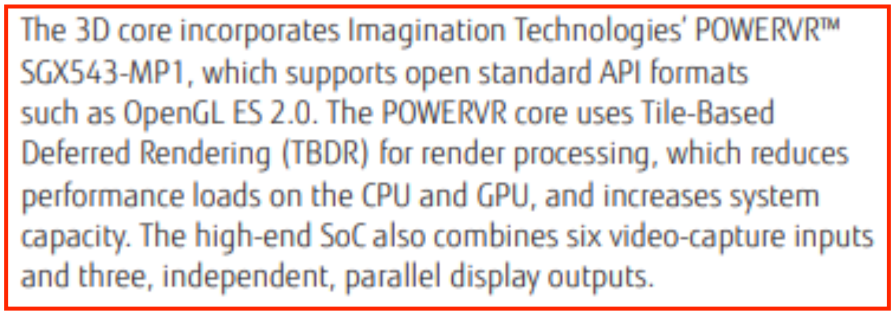
90. On information and belief, the Accused Instrumentalities necessarily infringe one or more claims of the ’478 Patent when used as intended.

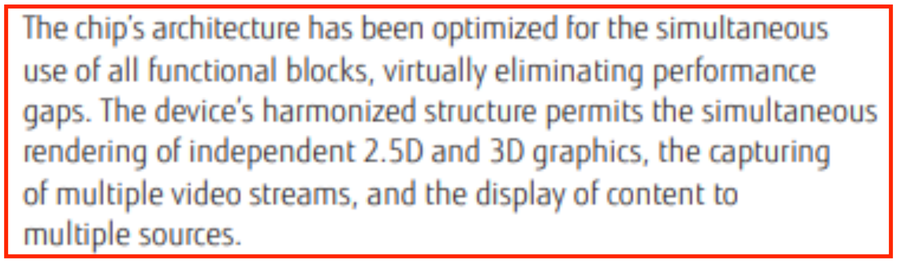
91. On information and belief, the Accused Instrumentalities infringe and induce others to infringe the ’478 Patent by providing a method for performing an arbitrary permutation of a source sequence of bits in a programmable processor. For example, Fujitsu provides a system-on-chip (including but not limited to MB86R1x, MB86R2x and MB86HD6x) solutions for parallel data processing. Defendant’s MB86R24 Triton-C SoC (used herein as an exemplary product) is used in Graphic Display Controllers. The MB86R24 Triton-C includes ARM Cortex-A9 dual CPU core, embedded 2.5D and 3D graphics cores. Further, the MB86R24 Triton-C SoC (“programmable processor”) utilizes Arm Neon technology (an advanced Single Instruction Multiple Data (SIMD) architecture) for improving audio/video encoding and decoding, 2.5D/3D graphics (“source sequence of bits”), and/or image/video processing. ARM Neon SIMD architecture provides permutation instructions to rearrange individual elements present in 2.5D/3D graphics.

92. For example, Defendant's 88PA6270 SoC (used herein as an exemplary product) is used for class color and monochrome single or multi-function printers. The 88PA6270 SoC includes a quad core 1.2 GHz ARM A53 processor to handle all the application processing and Page Description Language (PDL) rendering requirements.

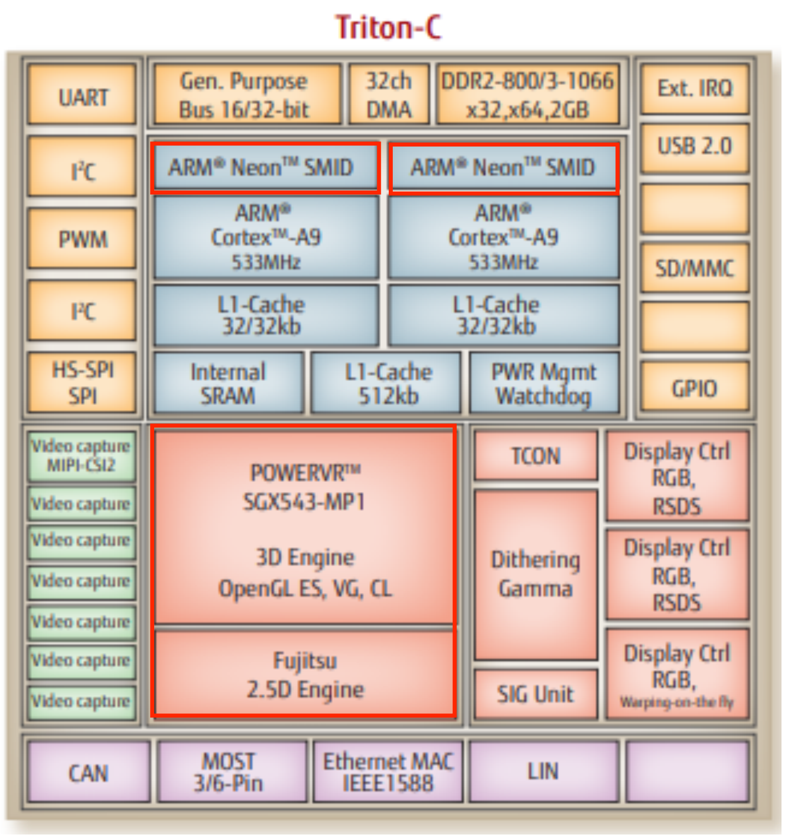
93. Further, the 88PA6270 SoC ("programmable processor") utilizes ARM Neon technology (a Single Instruction Multiple Data (SIMD) architecture) for improving video encoding and decoding, 2D/3D graphics ("two dimensional (2-D) data"), and/or gaming experience. ARM Neon SIMD architecture provides permutation instructions to rearrange individual elements present in 2D/3D graphics.


MB86R24 "Triton" Series: The high-performance MB86R24 "Triton-C" combines the latest ARM Cortex-A9 dual CPU core with state-of-the-art, embedded 2.5D and 3D graphics cores. This third-generation application processor is the first device in Fujitsu's new "Blueline" family of high-performance GDCs.

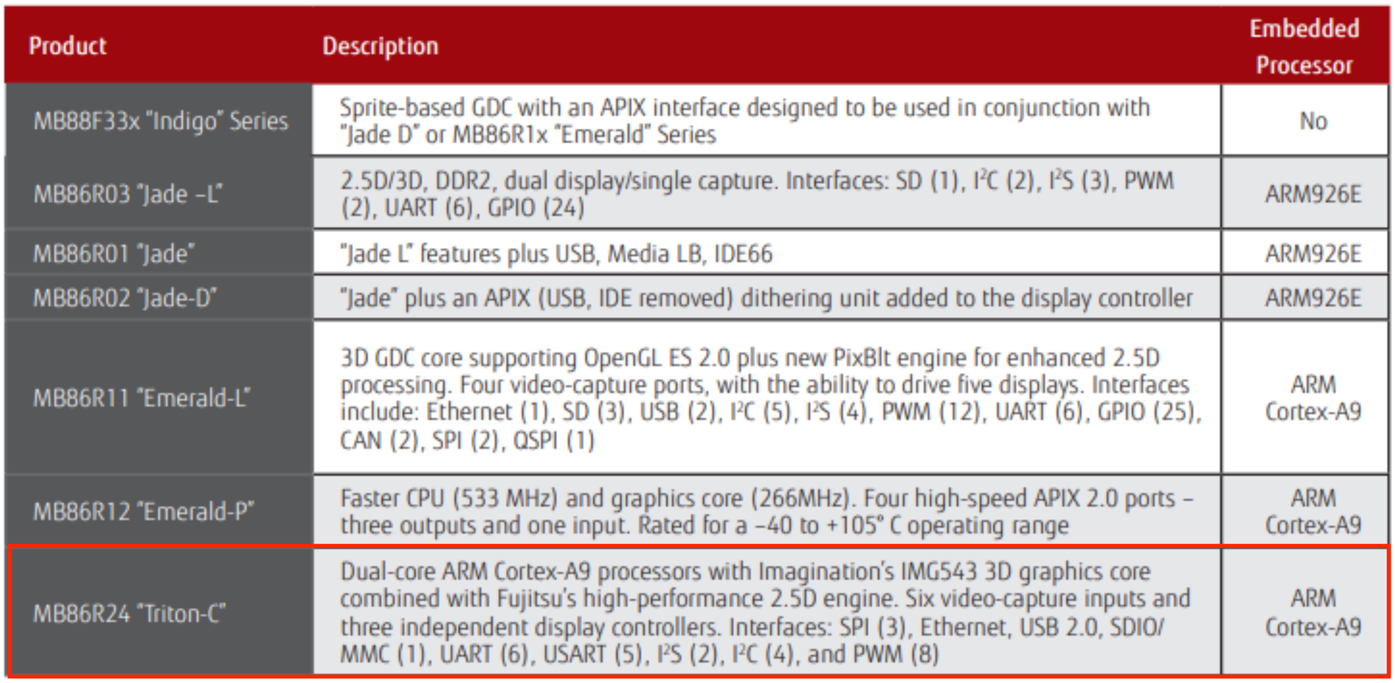

The 3D core incorporates Imagination Technologies' POWERVR™ SGX543-MP1, which supports open standard API formats such as OpenGL ES 2.0. The POWERVR core uses Tile-Based Deferred Rendering (TBDR) for render processing, which reduces performance loads on the CPU and GPU, and increases system capacity. The high-end SoC also combines six video-capture inputs and three, independent, parallel display outputs.


The chip's architecture has been optimized for the simultaneous use of all functional blocks, virtually eliminating performance gaps. The device's harmonized structure permits the simultaneous rendering of independent 2.5D and 3D graphics, the capturing of multiple video streams, and the display of content to multiple sources.

Source:https://www.fujitsu.com/us/Images/SPBG_GDC_Overview_PB.pdf, page 3

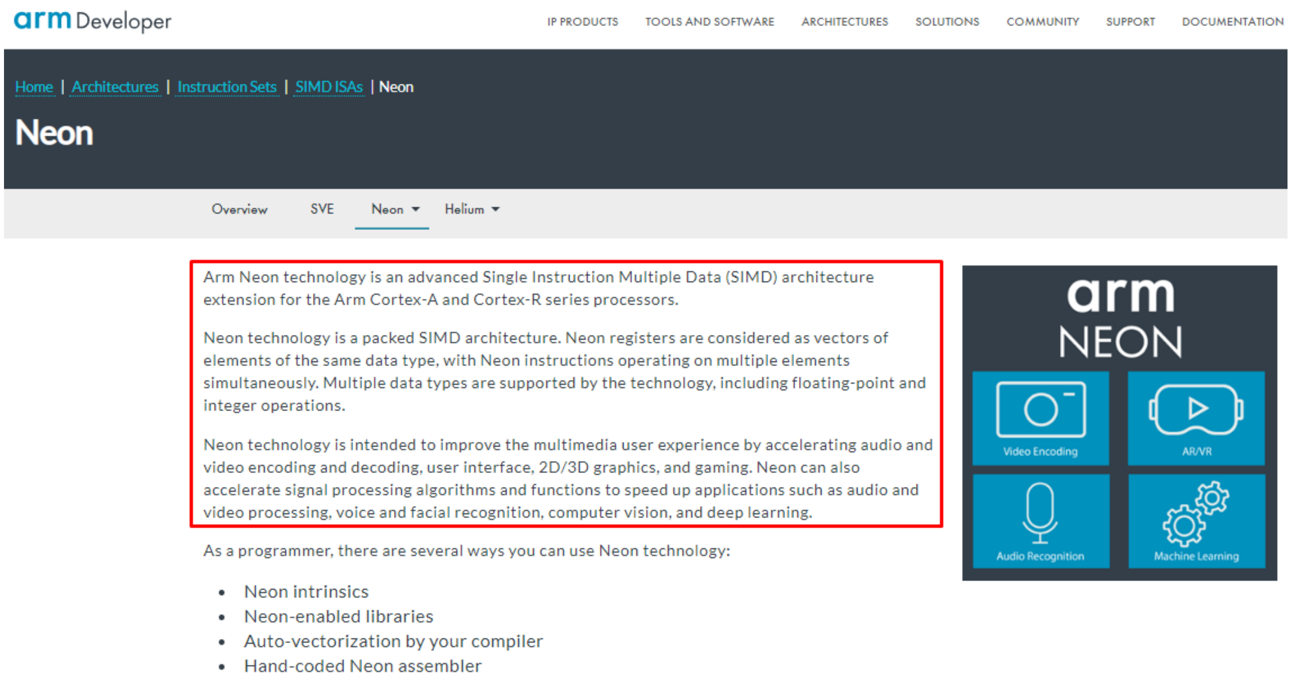


Source:https://www.fujitsu.com/us/Images/SPBG_GDC_Overview_PB.pdf, page 3



Product	Description	Embedded Processor
MB88F33x "Indigo" Series	Sprite-based GDC with an APIX interface designed to be used in conjunction with "Jade D" or MB86R1x "Emerald" Series	No
MB86R03 "Jade -L"	2.5D/3D, DDR2, dual display/single capture. Interfaces: SD (1), I²C (2), I²S (3), PWM (2), UART (6), GPIO (24)	ARM926E
MB86R01 "Jade"	"Jade L" features plus USB, Media LB, IDE66	ARM926E
MB86R02 "Jade-D"	"Jade" plus an APIX (USB, IDE removed) dithering unit added to the display controller	ARM926E
MB86R11 "Emerald-L"	3D GDC core supporting OpenGL ES 2.0 plus new PixBlt engine for enhanced 2.5D processing. Four video-capture ports, with the ability to drive five displays. Interfaces include: Ethernet (1), SD (3), USB (2), I²C (5), I²S (4), PWM (12), UART (6), GPIO (25), CAN (2), SPI (2), QSPI (1)	ARM Cortex-A9
MB86R12 "Emerald-P"	Faster CPU (533 MHz) and graphics core (266MHz). Four high-speed APIX 2.0 ports – three outputs and one input. Rated for a –40 to +105° C operating range	ARM Cortex-A9
MB86R24 "Triton-C"	Dual-core ARM Cortex-A9 processors with Imagination's IMG543 3D graphics core combined with Fujitsu's high-performance 2.5D engine. Six video-capture inputs and three independent display controllers. Interfaces: SPI (3), Ethernet, USB 2.0, SDIO/MMC (1), UART (6), USART (5), I²S (2), I²C (4), and PWM (8)	ARM Cortex-A9

Source:https://www.fujitsu.com/us/Images/SPBG_GDC_Overview_PB.pdf, page 4.



arm Developer

IP PRODUCTS TOOLS AND SOFTWARE ARCHITECTURES SOLUTIONS COMMUNITY SUPPORT DOCUMENTATION

Home | Architectures | Instruction Sets | SIMD ISAs | Neon

Neon

Overview SVE Neon Helium

Arm Neon technology is an advanced Single Instruction Multiple Data (SIMD) architecture extension for the Arm Cortex-A and Cortex-R series processors.

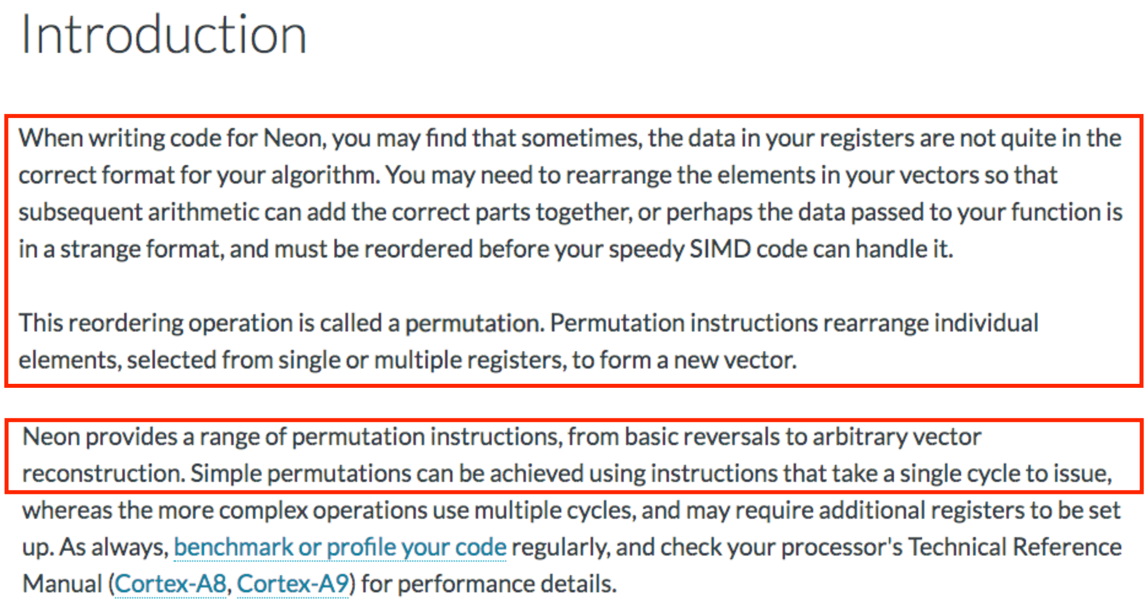
Neon technology is a packed SIMD architecture. Neon registers are considered as vectors of elements of the same data type, with Neon instructions operating on multiple elements simultaneously. Multiple data types are supported by the technology, including floating-point and integer operations.

Neon technology is intended to improve the multimedia user experience by accelerating audio and video encoding and decoding, user interface, 2D/3D graphics, and gaming. Neon can also accelerate signal processing algorithms and functions to speed up applications such as audio and video processing, voice and facial recognition, computer vision, and deep learning.

As a programmer, there are several ways you can use Neon technology:

- Neon intrinsics
- Neon-enabled libraries
- Auto-vectorization by your compiler
- Hand-coded Neon assembler

Source: https://developer.arm.com/architectures/instruction-sets/simd-isas/neon



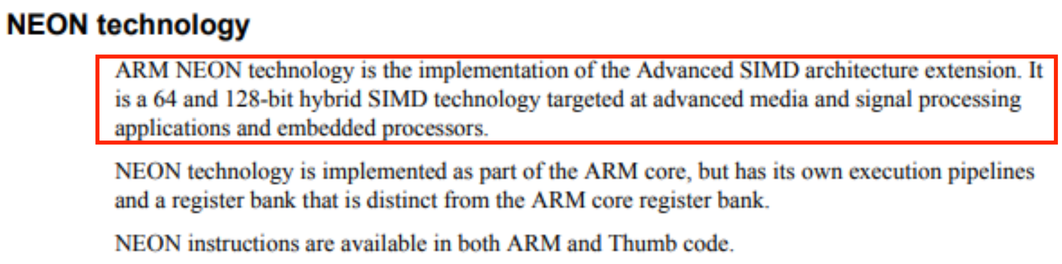
Introduction

When writing code for Neon, you may find that sometimes, the data in your registers are not quite in the correct format for your algorithm. You may need to rearrange the elements in your vectors so that subsequent arithmetic can add the correct parts together, or perhaps the data passed to your function is in a strange format, and must be reordered before your speedy SIMD code can handle it.

This reordering operation is called a permutation. Permutation instructions rearrange individual elements, selected from single or multiple registers, to form a new vector.

Neon provides a range of permutation instructions, from basic reversals to arbitrary vector reconstruction. Simple permutations can be achieved using instructions that take a single cycle to issue, whereas the more complex operations use multiple cycles, and may require additional registers to be set up. As always, benchmark or profile your code regularly, and check your processor's Technical Reference Manual (Cortex-A8, Cortex-A9) for performance details.

Source: https://community.arm.com/developer/ip-products/processors/b/processors-ip-blog/posts/coding-for-neon---part-5-rearranging-vectors


NEON technology

ARM NEON technology is the implementation of the Advanced SIMD architecture extension. It is a 64 and 128-bit hybrid SIMD technology targeted at advanced media and signal processing applications and embedded processors.

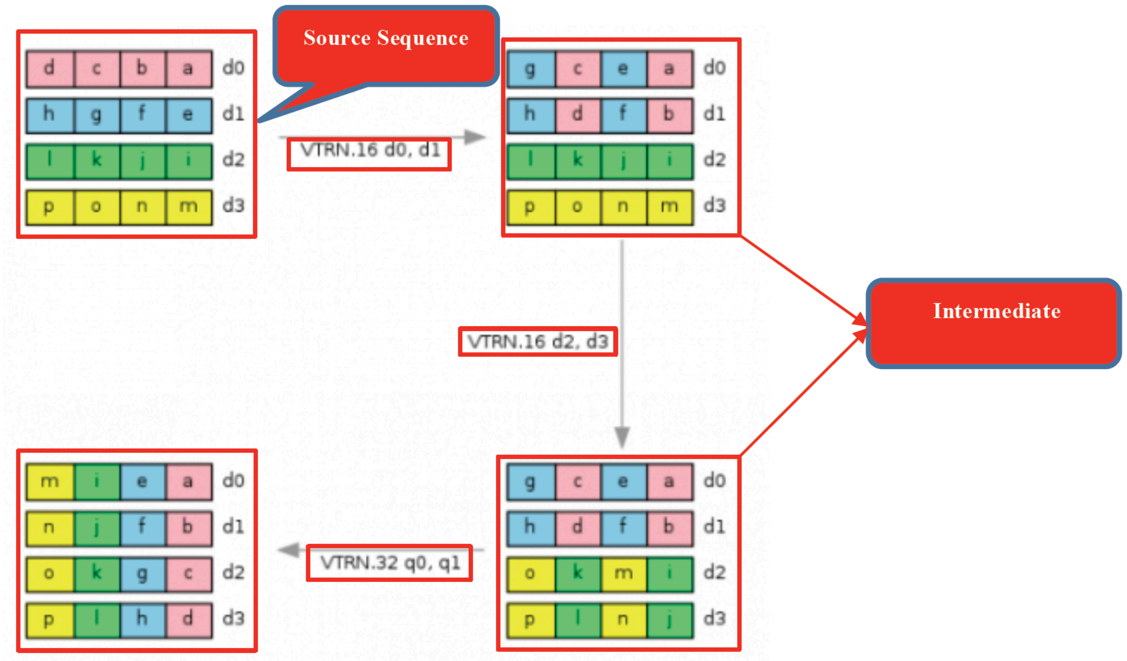
NEON technology is implemented as part of the ARM core, but has its own execution pipelines and a register bank that is distinct from the ARM core register bank.

NEON instructions are available in both ARM and Thumb code.

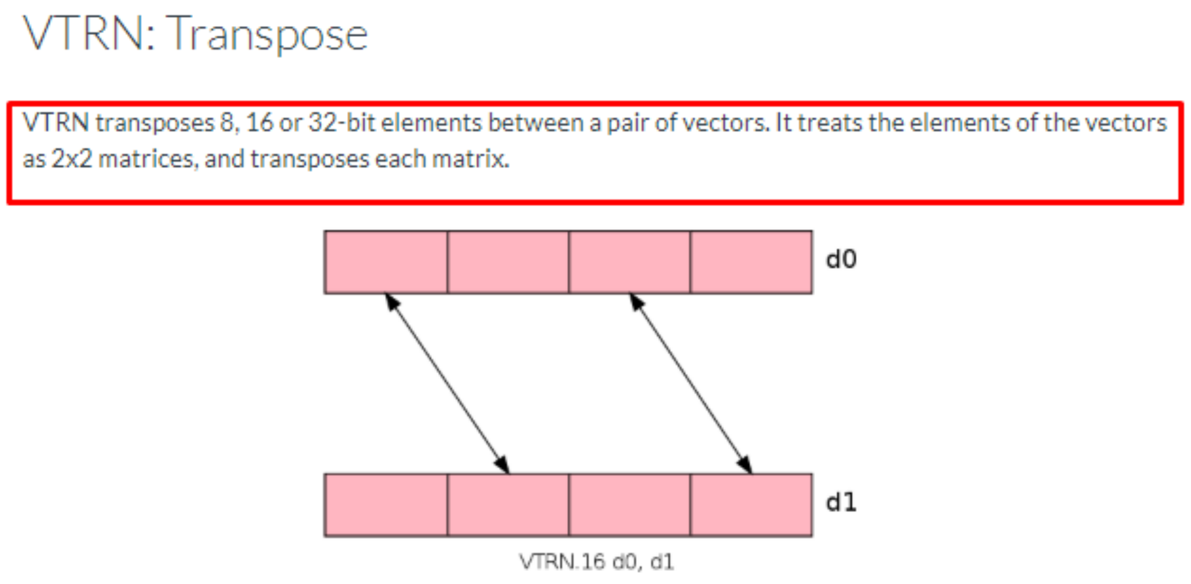
Source:

http://infocenter.arm.com/help/topic/com.arm.doc.dui0473j/DUI0473J_armasm_user_guide.pdf, page 2-40.

94. Further, Defendant performs the step of defining an intermediate sequence of bits that said source sequence of bits is transformed into. For example, the MB86R24 Triton-C SoC uses a permutation instruction (such as a VTRN instruction) and decomposes two dimensional data (in the form of 4x4 matrix) into an intermediate sequence and then into a 2x2 matrix. The permutation instruction transposes ("rearrangement") 8, 16 or 32-bit elements between a pair of vectors. The permutation instruction is applied on the elements of the vectors by dividing it into 2x2 matrices. The intermediate sequence of bits is converted into the desired output by applying the permutation instruction on it.



Source: https://community.arm.com/developer/ip-products/processors/b/processors-ip-blog/posts/coding-for-neon---part-5-rearranging-vectors (annotated)

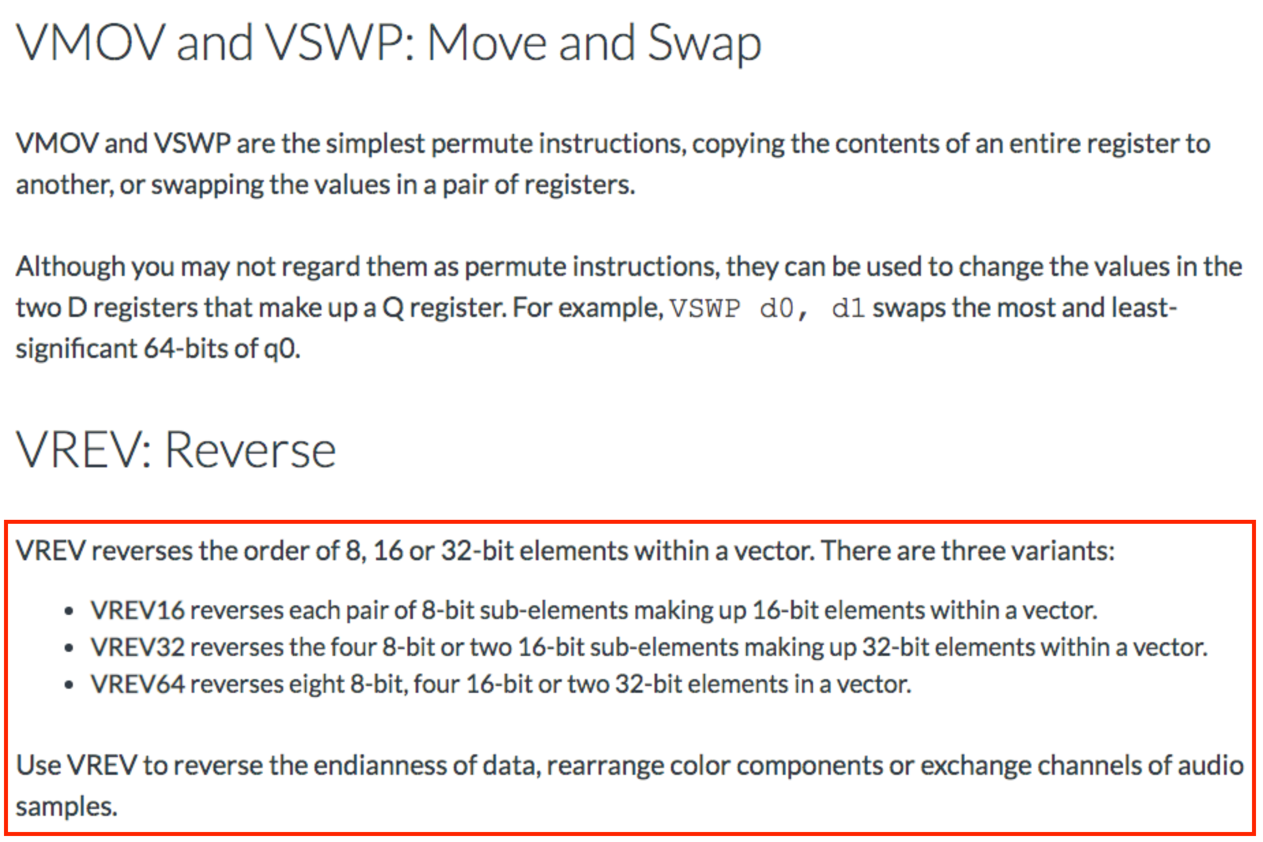


VTRN: Transpose

VTRN transposes 8, 16 or 32-bit elements between a pair of vectors. It treats the elements of the vectors as 2x2 matrices, and transposes each matrix.

Source: https://community.arm.com/developer/ip-products/processors/b/processors-ip-blog/posts/coding-for-neon---part-5-rearranging-vectors

95. Further, Defendant performs the step of determining a permutation instruction for transforming said source sequence of bits into said intermediate sequence of bits. For example, the MB86R24 Triton-C SoC supports multiple permutation instructions (including but not limited to VMOV, VSWP, VREV, VEXT, VTRN, VZIP, VUZP, VTBL, and VTBX) to carry out the permutation on the sequence of bits. The VTRN instruction is determined and applied on the source sequence of bits which transposes ("transforms") it into an intermediate sequence of bits on which further permutation instruction is applied to achieve the final permuted result.



VMOV and VSWP: Move and Swap

VMOV and VSWP are the simplest permute instructions, copying the contents of an entire register to another, or swapping the values in a pair of registers.

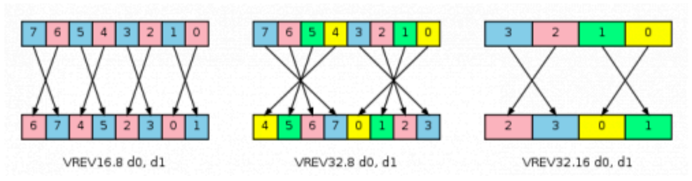
Although you may not regard them as permute instructions, they can be used to change the values in the two D registers that make up a Q register. For example, `VSWP d0, d1` swaps the most and least-significant 64-bits of q0.

VREV: Reverse

VREV reverses the order of 8, 16 or 32-bit elements within a vector. There are three variants:

- VREV16 reverses each pair of 8-bit sub-elements making up 16-bit elements within a vector.
- VREV32 reverses the four 8-bit or two 16-bit sub-elements making up 32-bit elements within a vector.
- VREV64 reverses eight 8-bit, four 16-bit or two 32-bit elements in a vector.

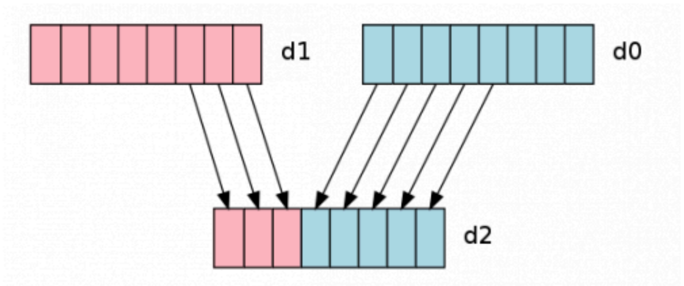
Use VREV to reverse the endianness of data, rearrange color components or exchange channels of audio samples.



VEXT: Extract

VEXT extracts a new vector of bytes from a pair of existing vectors. The bytes in the new vector are from the top of the first operand, and the bottom of the second operand. This allows you to produce a new vector containing elements that straddle a pair of existing vectors.

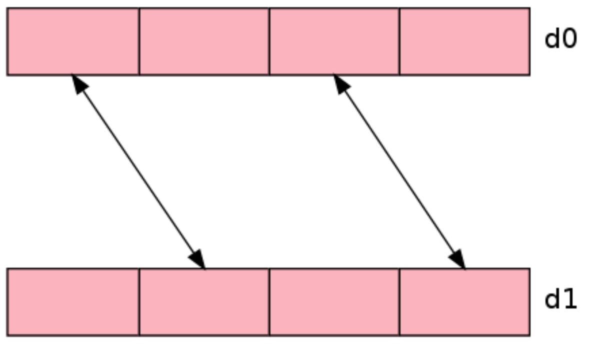
VEXT can be used to implement a moving window on data from two vectors, useful in FIR filters. For permutation, it can also be used to simulate a byte-wise rotate operation, when using the same vector for both input operands.



VEXT d2, d0, d1, #3

VTRN: Transpose

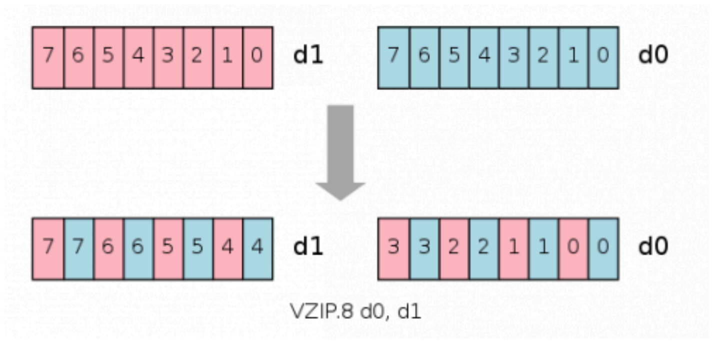
VTRN transposes 8, 16 or 32-bit elements between a pair of vectors. It treats the elements of the vectors as 2x2 matrices, and transposes each matrix.



VTRN.16 d0, d1

VZIP and VUZP: Zip and Unzip

VZIP interleaves the 8, 16 or 32-bit elements of a pair of vectors. The operation is the same as that performed by VST2 before storing, so use VST2 rather than VZIP if you need to zip data immediately before writing back to memory.



VZIP.8 d0, d1

VUZP is the inverse of VZIP, deinterleaving the 8, 16, or 32-bit elements of a pair of vectors. The operation is the same as that performed by VLD2 after loading from memory.

VTBL, VTBX: Table and Table Extend

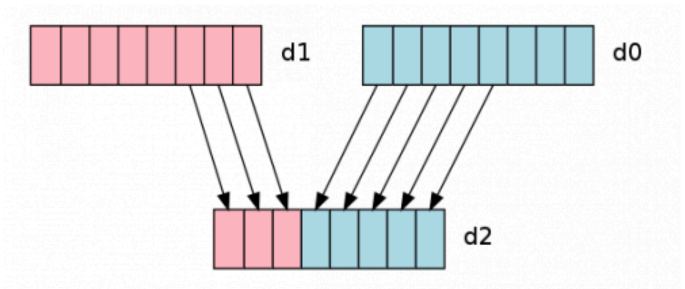
VTBL constructs a new vector from a table of vectors and an index vector. It is a byte-wise table lookup operation.

The table consists of one to four adjacent D registers. Each byte in the index vector is used to index a byte in the table of vectors. The indexed value is inserted into the result vector at the position corresponding to the location of the original index in the index vector.

VEXT: Extract

VEXT extracts a new vector of bytes from a pair of existing vectors. The bytes in the new vector are from the top of the first operand, and the bottom of the second operand. This allows you to produce a new vector containing elements that straddle a pair of existing vectors.

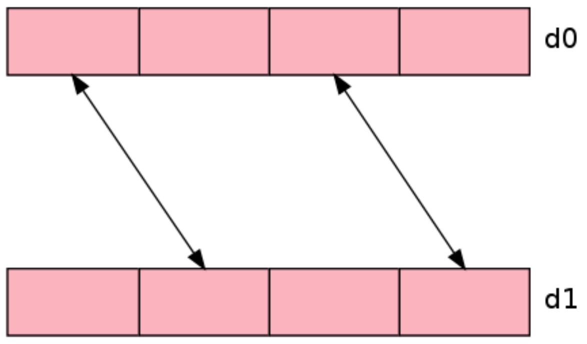
VEXT can be used to implement a moving window on data from two vectors, useful in FIR filters. For permutation, it can also be used to simulate a byte-wise rotate operation, when using the same vector for both input operands.



VEXT d2, d0, d1, #3

VTRN: Transpose

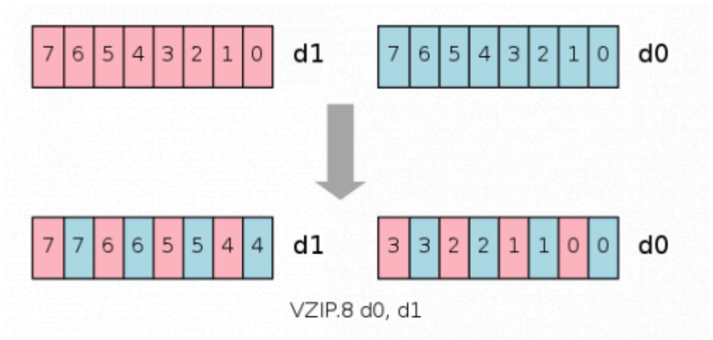
VTRN transposes 8, 16 or 32-bit elements between a pair of vectors. It treats the elements of the vectors as 2x2 matrices, and transposes each matrix.



VTRN.16 d0, d1

VZIP and VUZP: Zip and Unzip

VZIP interleaves the 8, 16 or 32-bit elements of a pair of vectors. The operation is the same as that performed by VST2 before storing, so use VST2 rather than VZIP if you need to zip data immediately before writing back to memory.



VUZP is the inverse of VZIP, deinterleaving the 8, 16, or 32-bit elements of a pair of vectors. The operation is the same as that performed by VLD2 after loading from memory.

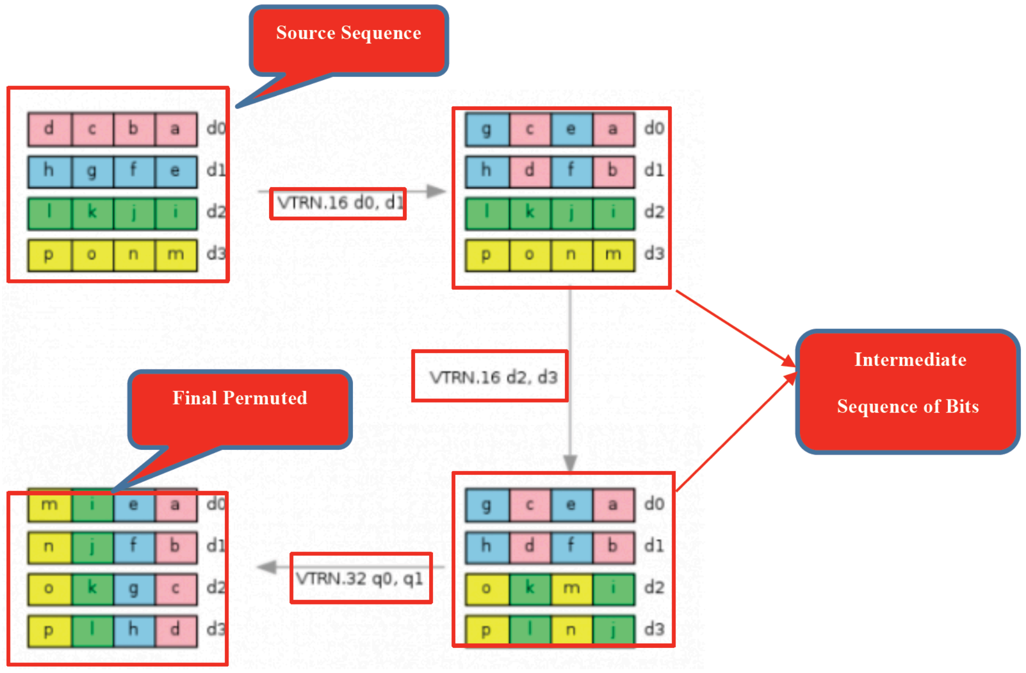
VTBL, VTBX: Table and Table Extend

VTBL constructs a new vector from a table of vectors and an index vector. It is a byte-wise table lookup operation.

The table consists of one to four adjacent D registers. Each byte in the index vector is used to index a byte in the table of vectors. The indexed value is inserted into the result vector at the position corresponding to the location of the original index in the index vector.

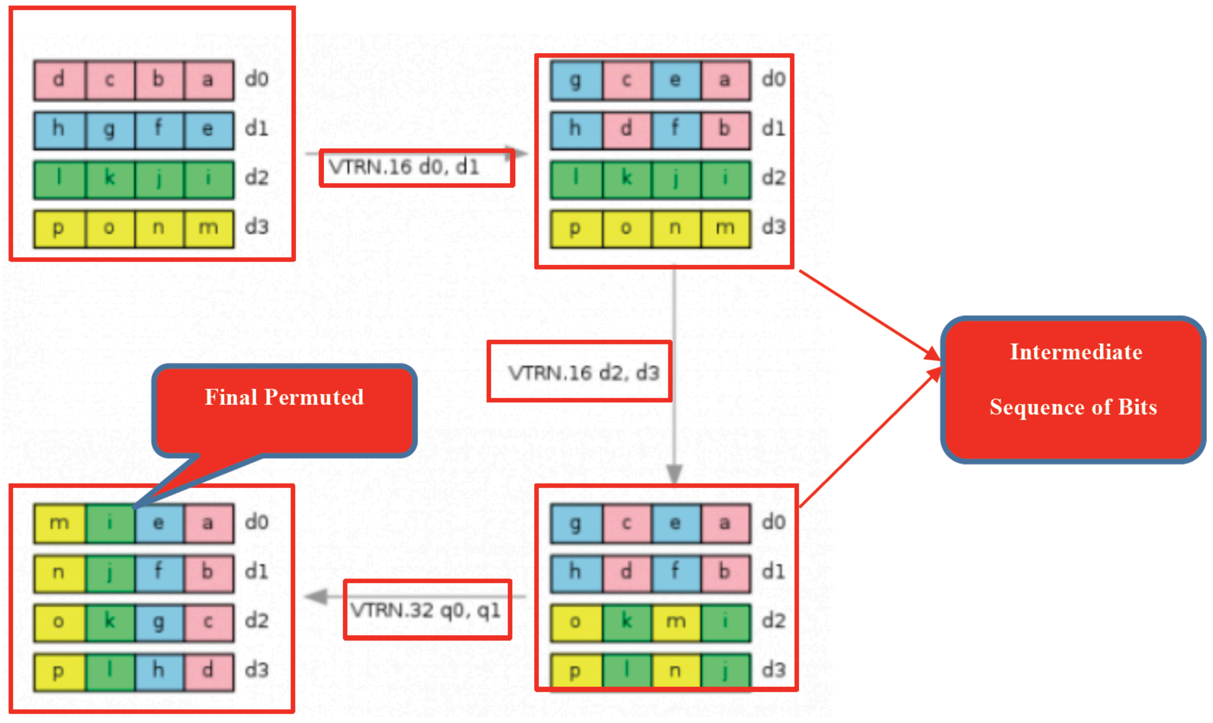
VTBL and VTBX differ in the way that out-of-range indexes are handled. If an index exceeds the length of the table, VTBL inserts zero at the corresponding position in the result vector, but VTBX leaves the value in the result vector unchanged.

If you use a single source vector as the table, VTBL allows you to implement an arbitrary permutation of a vector, at the expense of setting up an index register. If the operation is used in a loop, and the type of permutation doesn't change, you can initialize the index register outside the loop, and remove the setup overhead.



Source: https://community.arm.com/developer/ip-products/processors/b/processors-ip-blog/posts/coding-for-neon---part-5-rearranging-vectors (annotated).

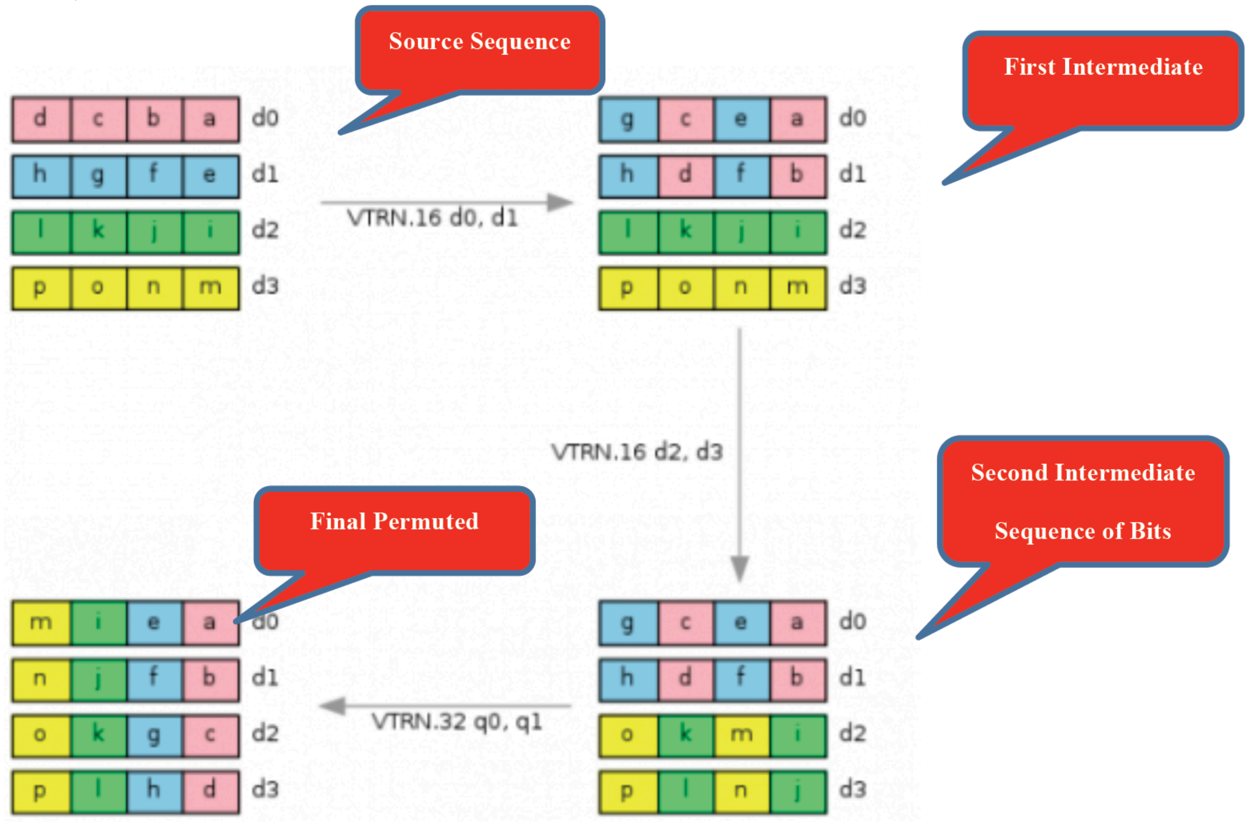
96. Further, Defendant performs the step of repeating steps (a.) and (b.) using said determined intermediate sequence of bits from step (b.) as said source sequence of bits in step (a.) until a desired sequence of bits is obtained. For example, the MB86R24 Triton-C SoC uses the permutation instruction to apply permutation on the intermediate sequence of bits until the desired permuted result ("sequence of bits") is obtained.



Source: https://community.arm.com/developer/ip-products/processors/b/processors-ip-blog/posts/coding-for-neon---part-5-rearranging-vectors (annotated).

97. Further, Defendant performs the step wherein the determined permutation instructions form a permutation instruction sequence. For example, first permutation instruction works as a source operand for the next instruction. Once the result value of the operation is written in the register file the result of the instruction is available to other instruction as a source operand and this is followed until the desired permuted result is obtained. The MB86R24 Triton-C SoC performs permutation of the source sequence of bits using VTRN, 16, d0, d1 ("permutation instruction") to obtain a first intermediate sequence of bits. Permutation instruction VTRN, 16, d2, d3 is applied on first intermediate sequence of bits further to obtain a second intermediate sequence of bits. VTRN, 16, q0, q1 is applied on second intermediate sequence of bits to get the final permuted result. The instructions (VTRN, 16, d0, d1) (VTRN, 16, d2, d3) and (VTRN, 16,

q0, q1) form sequence of permutation operations ("permutation instruction sequence") to permute the source sequence of bits.



Source: https://community.arm.com/developer/ip-products/processors/b/processors-ip-blog/posts/coding-for-neon---part-5-rearranging-vectors (annotated)

Cycles	This is the number of issue cycles the particular instruction consumes, and is the absolute minimum number of cycles per instruction if no operand interlocks are present.
Source	The Source field indicates the execution cycle where the source operands must be available if the instruction is to be permitted to issue. The comma separated list matches that of the Format field, indicating the register that each value applies to. Where two lines are provided for a single format containing quad (Q) registers, this indicates that the source registers are handled independently between top and bottom half double (D) registers. The first line provides the information for the lower half of the quad register and the second line provides the same information for the upper half of the quad register.
Result	The Result field indicates the execution cycle when the result of the operation is ready. At this point, the result might be ready as source operands for consumption by other instructions using forwarding paths. However, some pairs of instructions might have to wait until the value is written back to the register file.
Writeback	The Writeback field indicates the execution cycle that the result is committed to the register file. From this cycle, the result of the instruction is available to all other instructions as a source operand.

Source: https://static.docs.arm.com/ddi0409/i/DDI0409I_cortex_a9_neon_mpe_r4p1_trm.pdf , page 3-9.

98. Further, upon information and belief, Defendant directly infringes the claim at least when it tests its SoCs. During such tests, Defendant utilizes the SoCs to perform permutation on the input data using permutation instructions available in ARM Neon SIMD ISA (Instruction Set Architecture).

99. Further, Defendant indirectly infringes the claim at least when Defendant's customers (such as device manufacturers which use Defendant's SoCs in their products) perform the method while testing their devices and when the devices are operated by end-users.

100. Teleputers has been damaged by Defendant's infringement of the '478 Patent.

COUNT III
(Infringement of U.S. Patent No. 7,174,014)

101. Teleputers incorporates the above paragraphs by reference.

102. Defendant has been on notice of the ’014 Patent at least as early as the date it received service of this Original Complaint.

103. On information and belief, Defendant has infringed and continue to infringe the ’014 Patent by making, using, importing, selling, and/or, offering for sale the ’014 Accused Instrumentalities in the United States.

104. On information and belief, Defendant, with knowledge of the ’014 Patent, indirectly infringes the ’014 Patent by inducing others to infringe the ’014 Patent. In particular, Defendant intends to induce customers to infringe the ’014 Patent by encouraging customers to use the ’014 Accused Instrumentalities in a manner that results in infringement.

105. On information and belief, Defendant also induces others, including customers, to infringe the ’014 Patent by providing technical support for the use of the ’014 Accused Instrumentalities.

106. On information and belief, at all times Defendant owns and controls the operation of the ’014 Accused Instrumentalities in accordance with an end user license agreement.

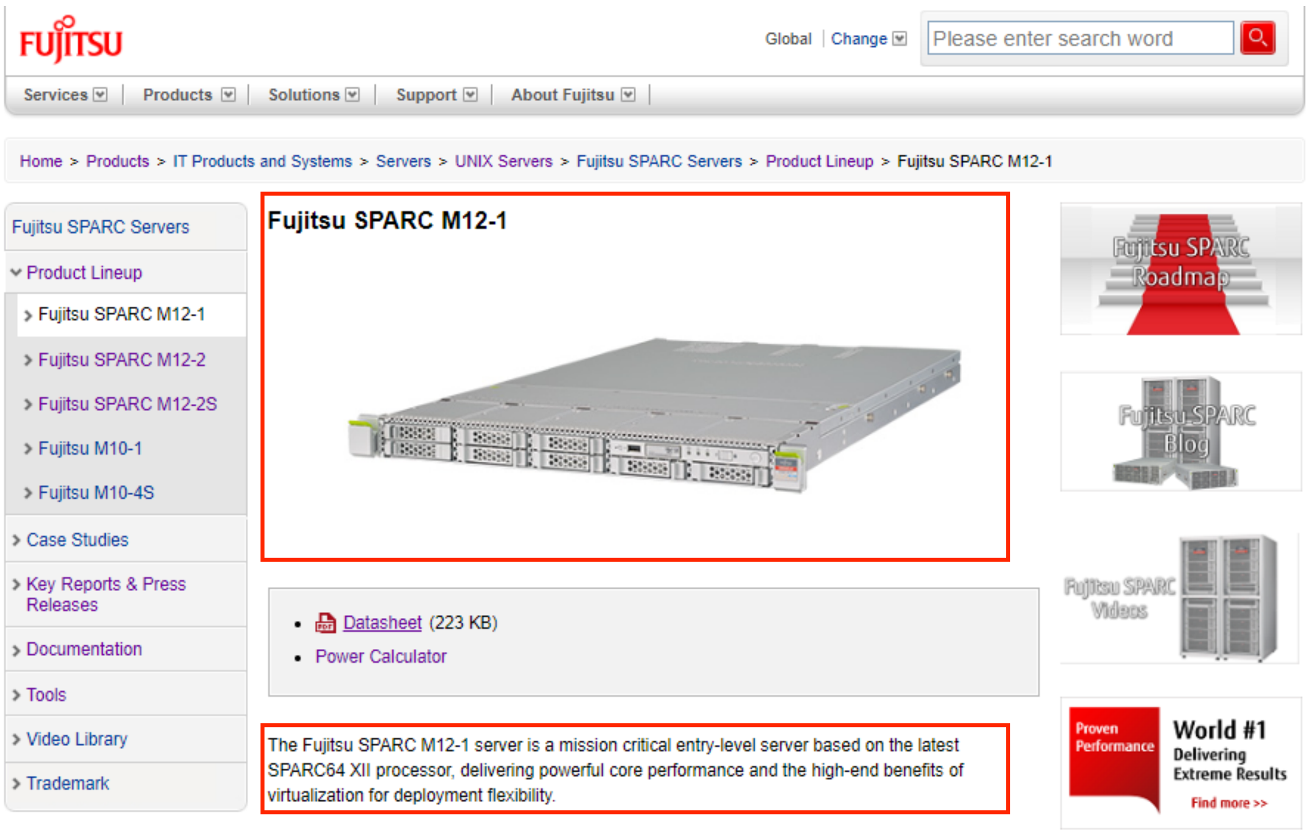
107. On information and belief, the Accused Instrumentalities necessarily infringe one or more claims of the ’014 Patent when used as intended.

108. On information and belief, the ’014 Accused Instrumentalities infringe and induce others to infringe the ’014 Patent by providing a method for performing an arbitrary permutation of a sequence of bits. For example, Defendant provides a SPARC64 XII processor used in Fujitsu servers (including but not limited to SPARC M12-1, SPARC M12-2 and SPARC M12-2S) for parallel data processing. Defendant’s SPARC M12-1 (used herein as an exemplary product) supports SPARC64 XII processor for mission-critical enterprise workloads and cloud computing. SPARC64 XII supports permutation instructions to permute the data in the register.

Fujitsu SPARC Servers Product Lineup

Models	Type	CPU	Max CPUs	Max Cores	Max Threads	Max Memory	Max Internal HDD
Fujitsu SPARC M12-1	Rackmount (1U)	SPARC64™ XII 3.2GHz	1	6	48	1TB	7.2TB
Fujitsu SPARC M12-2	Rackmount (4U)	SPARC64™ XII 3.9GHz	2	24	192	2TB	7.2TB
Fujitsu SPARC M12-2S	Rackmount (4U)	SPARC64™ XII 4.25GHz	2	24	192	2TB	7.2TB
	Two dedicated racks		32	384	3072	32TB	115.2TB

Source: https://www.fujitsu.com/us/products/computing/servers/unix/sparc/lineup/



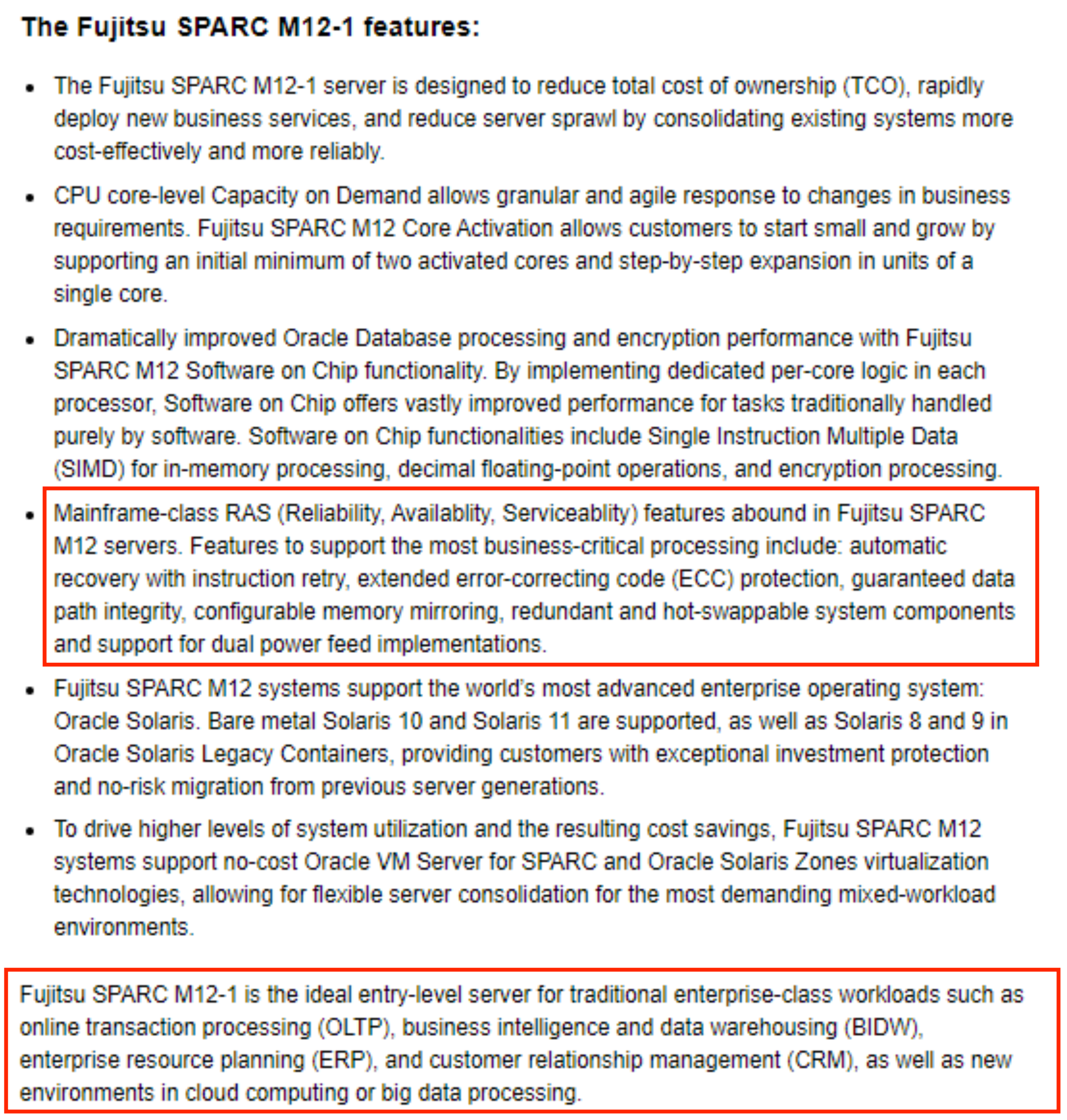
Home > Products > IT Products and Systems > Servers > UNIX Servers > Fujitsu SPARC Servers > Product Lineup > Fujitsu SPARC M12-1

Fujitsu SPARC M12-1

- Datasheet (223 KB)
- Power Calculator

The Fujitsu SPARC M12-1 server is a mission critical entry-level server based on the latest SPARC64 XII processor, delivering powerful core performance and the high-end benefits of virtualization for deployment flexibility.

Source: https://www.fujitsu.com/global/products/computing/servers/unix/sparc/lineup/m12-1/

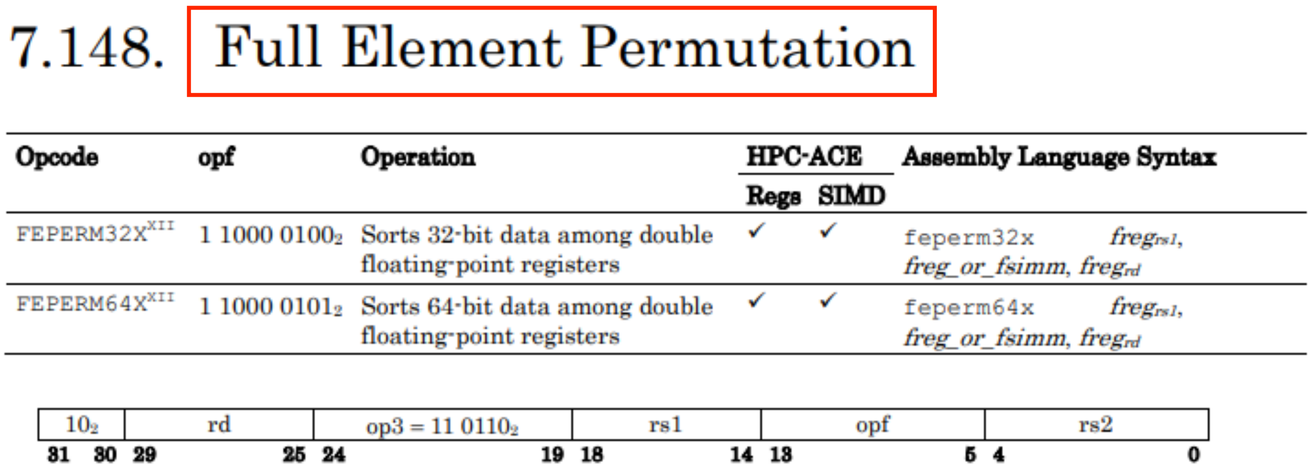


The Fujitsu SPARC M12-1 features:

- The Fujitsu SPARC M12-1 server is designed to reduce total cost of ownership (TCO), rapidly deploy new business services, and reduce server sprawl by consolidating existing systems more cost-effectively and more reliably.
- CPU core-level Capacity on Demand allows granular and agile response to changes in business requirements. Fujitsu SPARC M12 Core Activation allows customers to start small and grow by supporting an initial minimum of two activated cores and step-by-step expansion in units of a single core.
- Dramatically improved Oracle Database processing and encryption performance with Fujitsu SPARC M12 Software on Chip functionality. By implementing dedicated per-core logic in each processor, Software on Chip offers vastly improved performance for tasks traditionally handled purely by software. Software on Chip functionalities include Single Instruction Multiple Data (SIMD) for in-memory processing, decimal floating-point operations, and encryption processing.
- Mainframe-class RAS (Reliability, Availablity, Serviceablity) features abound in Fujitsu SPARC M12 servers. Features to support the most business-critical processing include: automatic recovery with instruction retry, extended error-correcting code (ECC) protection, guaranteed data path integrity, configurable memory mirroring, redundant and hot-swappable system components and support for dual power feed implementations.
- Fujitsu SPARC M12 systems support the world's most advanced enterprise operating system: Oracle Solaris. Bare metal Solaris 10 and Solaris 11 are supported, as well as Solaris 8 and 9 in Oracle Solaris Legacy Containers, providing customers with exceptional investment protection and no-risk migration from previous server generations.
- To drive higher levels of system utilization and the resulting cost savings, Fujitsu SPARC M12 systems support no-cost Oracle VM Server for SPARC and Oracle Solaris Zones virtualization technologies, allowing for flexible server consolidation for the most demanding mixed-workload environments.

Fujitsu SPARC M12-1 is the ideal entry-level server for traditional enterprise-class workloads such as online transaction processing (OLTP), business intelligence and data warehousing (BIDW), enterprise resource planning (ERP), and customer relationship management (CRM), as well as new environments in cloud computing or big data processing.

Source: https://www.fujitsu.com/global/products/computing/servers/unix/sparc/lineup/m12-1/



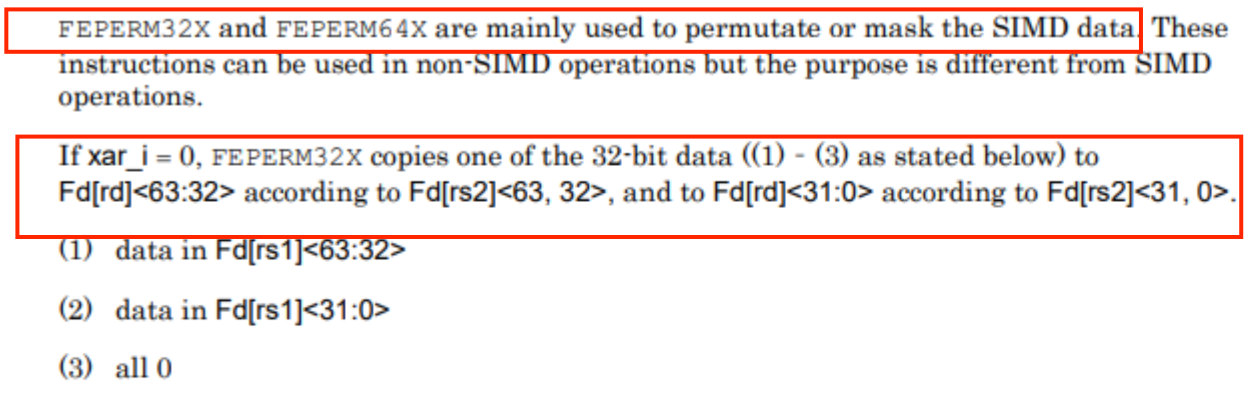
7.148. Full Element Permutation

Opcode	opf	Operation	HPC-ACE		Assembly Language Syntax
			Regs	SIMD	
FEPERM32X[XII]	1 1000 0100$_2$	Sorts 32-bit data among double floating-point registers	✓	✓	feperm32x *freg$_{rs1}$*, *freg_or_fsimm*, *freg$_{rd}$*
FEPERM64X[XII]	1 1000 0101$_2$	Sorts 64-bit data among double floating-point registers	✓	✓	feperm64x *freg$_{rs1}$*, *freg_or_fsimm*, *freg$_{rd}$*

10$_2$	rd	op3 = 11 0110$_2$	rs1	opf	rs2
31 30	29 25	24 19	18 14	13 5	4 0

Source:https://www.fujitsu.com/jp/documents/products/computing/servers/unix/sparc/downloads/documents/SPARC64XII-Specification.vol20.pdf page-75

109. Further, Defendant performs and induces others to perform the step of inputting a source sequence of bits into a source register. For example, the SPARC M12-1 uses a permutation instruction (such as a FEPERM32X/FEPERM64X instruction) to permute the data bits. Fd[rs1] ("source register") contains the data bits to be permuted.



FEPERM32X and FEPERM64X are mainly used to permutate or mask the SIMD data. These instructions can be used in non-SIMD operations but the purpose is different from SIMD operations.

If xar_i = 0, FEPERM32X copies one of the 32-bit data ((1) - (3) as stated below) to Fd[rd]<63:32> according to Fd[rs2]<63, 32>, and to Fd[rd]<31:0> according to Fd[rs2]<31, 0>.

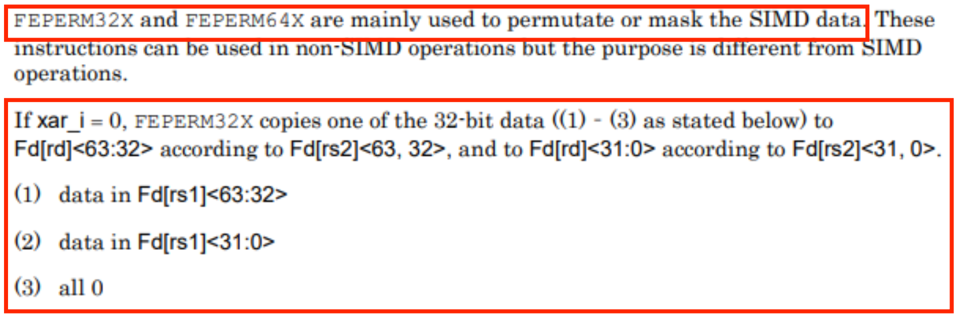
(1) data in Fd[rs1]<63:32>

(2) data in Fd[rs1]<31:0>

(3) all 0

Source:https://www.fujitsu.com/jp/documents/products/computing/servers/unix/sparc/downloads/documents/SPARC64XII-Specification.vol20.pdf, page-75.

110. Further, Defendant performs and induces others to perform the step of defining bit positions in said source sequence of bits to be permuted in said source register for a group of bits in a destination register. For example, the SPARC M12-1 implements a permutation instruction (such as a FEPERM32X/ FEPERM64X instruction) to permute the data bits. The data bits in the Fd[rs1] are permuted based on the data bits in Fd[rs2]. Therefore, Fd[rs2] determines the data bits in the Fd[rs1] that will be permuted and moved to the destination register Fd[rd]. If the 63rd bit of Fd[rs2] is 0 and the 32nd bit of the Fd[rs2] is 0, then the [63-32] bits of Fd[rs1] will be copied to [63-32] bit position in Fd[rd]. If the 63rd bit of Fd[rs2] is 0 and the 32nd bit of the Fd[rs2] is 1, then the [31-0] bits in Fd[rs1] will be copied to [63-32] bit position in Fd[rd]. Similarly, for the bit position [31-0] in Fd[rd], 31st and 0th bit of Fd[rs2] are checked.


FEPERM32X and FEPERM64X are mainly used to permutate or mask the SIMD data. These instructions can be used in non-SIMD operations but the purpose is different from SIMD operations.

If xar_i = 0, FEPERM32X copies one of the 32-bit data ((1) - (3) as stated below) to Fd[rd]<63:32> according to Fd[rs2]<63, 32>, and to Fd[rd]<31:0> according to Fd[rs2]<31, 0>.

(1) data in Fd[rs1]<63:32>

(2) data in Fd[rs1]<31:0>

(3) all 0

Source:https://www.fujitsu.com/jp/documents/products/computing/servers/unix/sparc/downloads/documents/SPARC64XII-Specification.vol20.pdf page-75.

The behavior of FEPERM32X is described in Figure 7-8, Table 7-10, and Table 7-11. The value of Fd[rs2]<62:33, 30:1> is ignored.

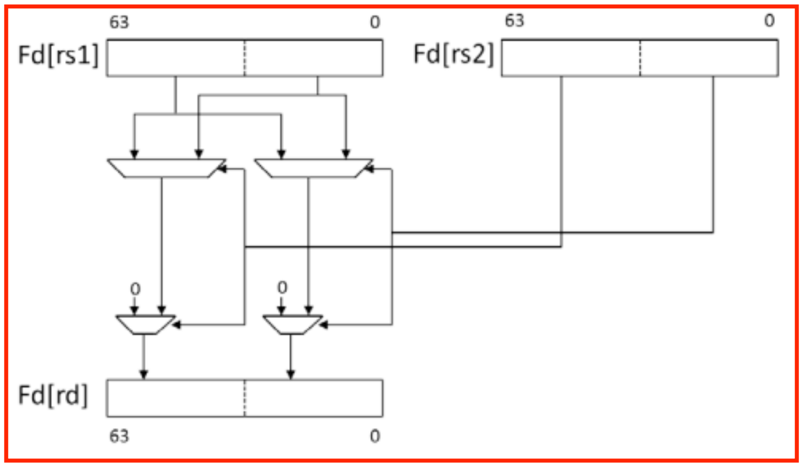


Figure 7-8 Behavior of FEPERM32X (xar_i = 0)

Source:https://www.fujitsu.com/jp/documents/products/computing/servers/unix/sparc/downloads /documents/SPARC64XII-Specification.vol20.pdf page-75

Table 7-10 Results of FEPERM32X (Fd[rd]<63:32>, xar_i = 0)

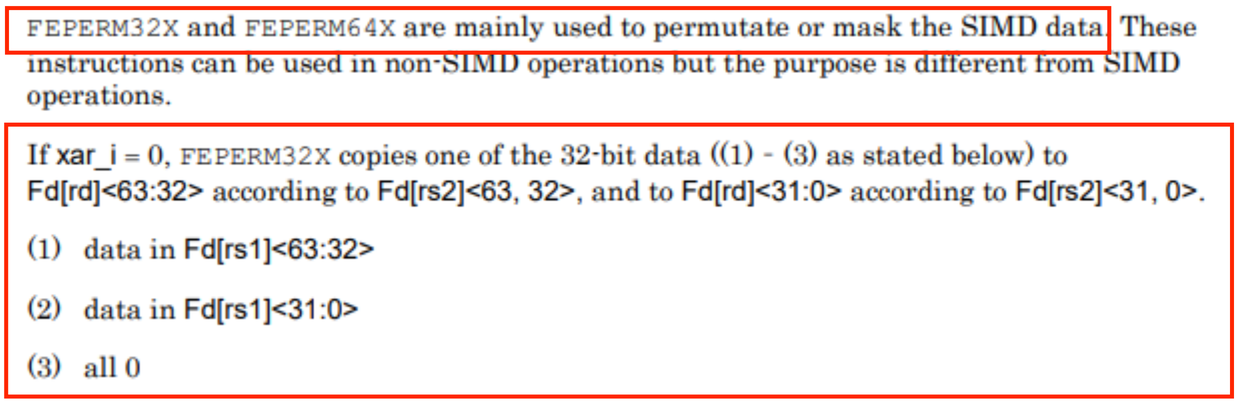
Fd[rs2]<63>	Fd[rs2]<32>	Fd[rd]<63:32>
0	0	Fd[rs1]<63:32>
	1	Fd[rs1]<31:0>
1	–	all 0

Table 7-11 Results of FEPERM32X (Fd[rd]<31:0>, xar_i = 0)

Fd[rs2]<31>	Fd[rs2]<0>	Fd[rd]<31:0>
0	0	Fd[rs1]<63:32>
	1	Fd[rs1]<31:0>
1	–	all 0

Source:https://www.fujitsu.com/jp/documents/products/computing/servers/unix/sparc/downloads /documents/SPARC64XII-Specification.vol20.pdf page-76.

111. Further, Defendant performs and induces others to perform the step of in response to a PPERM instruction inserting bits from said source sequence into said destination register as determined by said bit positions. For example, the SPARC M12-1 performs the FEPERM32X/FEPERM64X permutation instructions that are equivalent to the PPERM instruction as mentioned in the claimed invention. In response to the FEPERM32X/ FEPERM64X permutation instruction, the data bits in Fd[rs1] ("source sequence") are moved to the Fd[rd] ("destination register") according to the bit positions defined by Fd[rs2].


FEPERM32X and FEPERM64X are mainly used to permutate or mask the SIMD data. These instructions can be used in non-SIMD operations but the purpose is different from SIMD operations.

If xar_i = 0, FEPERM32X copies one of the 32-bit data ((1) - (3) as stated below) to Fd[rd]<63:32> according to Fd[rs2]<63, 32>, and to Fd[rd]<31:0> according to Fd[rs2]<31, 0>.

(1) data in Fd[rs1]<63:32>

(2) data in Fd[rs1]<31:0>

(3) all 0

Source:https://www.fujitsu.com/jp/documents/products/computing/servers/unix/sparc/downloads/documents/SPARC64XII-Specification.vol20.pdf page-75.

112. Further, upon information and belief, Defendant directly infringes the claim at least when it tests its servers. During such tests, Defendant utilizes the SPARC XII processor to perform permutation on the input data using permutation instructions.

113. Further, Further, Defendant indirectly infringes the claim at least when the server computers are operated by the customers. During such use, end users utilize the SPARC XII processor to perform permutation on the input data using permutation instructions.

114. Teleputers has been damaged by Defendant's infringement of the '014 Patent.

PRAYER FOR RELIEF

WHEREFORE, Teleputers respectfully requests the Court enter judgment against Defendants:

declaring that the Defendants have infringed each of the Patents-in-Suit;

a) awarding Teleputers its damages suffered as a result of Defendants' infringement of the Patents-in-Suit;

b) awarding Teleputers its costs, attorneys' fees, expenses, and interest;

c) awarding Teleputers ongoing post-trial royalties; and

d) granting Teleputers such further relief as the Court finds appropriate.

<u>JURY DEMAND</u>

Teleputers demands trial by jury, under Fed. R. Civ. P. 38.

Dated: October 7, 2020

Respectfully Submitted

/s/ ________________________

Randall Garteiser
Texas Bar No. 24038912
M. Scott Fuller
Texas Bar No. 24036607
sfuller@ghiplaw.com
Thomas G. Fasone III
Texas Bar No. 00785382
tfasone@ghiplaw.com
GARTEISER HONEA, PLLC
119 W. Ferguson Street
Tyler, Texas 75702
Telephone: (903) 705-7420
Facsimile: (888) 908-4400

Raymond W. Mort, III
Texas State Bar No. 00791308
raymort@austinlaw.com
THE MORT LAW FIRM, PLLC
100 Congress Ave, Suite 2000
Austin, Texas 78701
Tel/Fax: (512) 865-7950

**ATTORNEYS FOR PLAINTIFF
TELEPUTERS LLC**