

**UNITED STATES DISTRICT COURT
FOR THE EASTERN DISTRICT OF VIRGINIA
NORFOLK DIVISION**

HL KLEMOVE CORP.,

Plaintiff,

v.

FORAS TECHNOLOGIES LTD.,

Defendant.

Case No. 2:23-cv-00647-EWH-RJK

JURY TRIAL DEMANDED

**FORAS TECHNOLOGIES LTD.'S ANSWER, AFFIRMATIVE DEFENSES, AND
COUNTERCLAIMS**

Defendant Foras Technologies Ltd. ("Foras"), by and through its undersigned counsel, hereby responds to the Complaint for Declaratory Judgment of Plaintiff HL Klemove Corp. ("HLK") (Dkt. No. 1, "Complaint") with the following Answer, Affirmative Defenses, and Counterclaims.

GENERAL DENIAL

Foras responds to the allegations contained in the numbered paragraphs of HLK's Complaint. Except as specifically admitted herein, Foras denies all allegations and characterizations in HLK's Complaint.

RESPONSE TO SPECIFIC ALLEGATIONS

1. The allegations in paragraph 1 include HLK's characterization of the claims in this action to which no response is required. To the extent a response is required, Foras denies the allegations in paragraph 1.

NATURE OF THE ACTION¹

2. The allegations in paragraph 2 include HLK's characterization of the claims in this action to which no response is required. To the extent paragraph 2 implicates legal conclusions, no response is required. Foras admits that it has accused Kia Corporation and Kia America Inc. ("Kia") of infringement of U.S. Patent No. 7,502,958 ("the '958 Patent") through the sale of Kia-branded automobiles containing components sold by HLK and that Foras attempted to serve HLK with a subpoena in connection with that action. Foras lacks knowledge or information sufficient to form a belief as to the truth of the other allegations in paragraph 2, and therefore denies those allegations.

3. Foras lacks knowledge or information sufficient to form a belief as to the truth of the allegations in paragraph 3, and therefore denies those allegations.

4. Foras admits that it has filed a complaint against Kia Corporation and Kia America Inc. for infringement of the '958 Patent. Foras lacks knowledge or information sufficient to form a belief as to the truth of the allegations in paragraph 4, and therefore denies those allegations.

5. Foras admits that it is the sole owner by assignment of all right, title, and interest of the '958 Patent, including the right to recover for past, present, and future infringement. Foras further admits that the complaint in the case captioned *Foras Techs. Ltd. v. Kia Corp., et al.*, No. 2:23-cv-00219-JRG is attached as Exhibit 2 to HLK's Complaint and that the corresponding '958 Patent claim chart is attached as Exhibit 3 to HLK's Complaint. Foras denies the remaining allegations in paragraph 5.

¹ Foras repeats the headings in the Complaint. In doing so, Foras makes no admissions regarding the substance of the headings. Unless otherwise stated, to the extent that a particular heading can be construed as an allegation, Foras specifically denies all such allegations.

6. Foras lacks knowledge or information sufficient to form a belief as to the truth of the allegations in paragraph 6, and therefore denies those allegations.

7. Foras admits that the cases identified in the table at paragraph 7 were brought by Foras and involve the '958 Patent. Foras lacks knowledge or information sufficient to form a belief as to the truth of the other allegations in paragraph 7, and therefore denies those allegations.

8. Foras admits that it filed a complaint against Kia Corporation and Kia America Inc. alleging that certain Kia automobiles, including the Kia K5, Telluride, and Seltos, include an MRR-20 Mid-Range Radar (containing an Infineon SAK-TC297TA chipset) and/or a Hella RS4 Lane Change Assist Radar (containing an Infineon SAK-TC264DA chipset) (the "Accused Products") infringe at least claims 1, 2, 3, 6, 8, 19, and 20 of the '958 Patent. Foras lacks knowledge or information sufficient to form a belief as to the truth of the other allegations in paragraph 8, and therefore denies those allegations.

9. Foras lacks knowledge or information sufficient to form a belief as to the truth of the allegations in paragraph 9, and therefore denies those allegations.

10. Foras lacks knowledge or information sufficient to form a belief as to the truth of the allegations in paragraph 10, and therefore denies those allegations.

11. To the extent paragraph 11 implicates legal conclusions, no response is required. To the extent a response is required, Foras admits that HLK purports to seek a declaratory judgment, but denies any remaining allegations in paragraph 11.

12. To the extent paragraph 12 implicates legal conclusions, no response is required. To the extent a response is required, Foras admits that HLK purports to seek a declaratory judgment, but denies any remaining allegations in paragraph 12.

THE PARTIES

13. On information and belief, Foras admits that HLK is a corporation organized under the laws of the Republic of Korea with its principal place of business at 224 Harmony-ro, Yeonsu-gu, Incheon, Republic of Korea.

14. Admitted.

PATENT ASSERTED AGAINST PLAINTIFF'S CUSTOMERS

15. Foras admits that the '958 Patent relates to certain methods and systems for detecting and recovering from a loss of lockstep for a pair of processors. Foras admits that HLK's Accused Products infringe those methods and systems disclosed in the claims of the '958 Patent.

16. Foras lacks knowledge or information sufficient to form a belief as to the truth of the allegations in paragraph 16, and therefore denies those allegations.

17. Admitted.

18. Foras admits that HLK quotes from the '958 Patent at 3:50-55 and 3:55-59, but denies any remaining allegations in paragraph 18.

JURISDICTION AND VENUE

19. Foras repeats and incorporates by reference as if fully set forth herein its responses to paragraphs 1-18 above.

20. To the extent paragraph 20 of the Complaint implicates legal conclusions, no response is required. To the extent a response is required, Foras admits that this Court has subject matter jurisdiction over the Complaint, but denies any remaining allegations in paragraph 20.

21. To the extent paragraph 21 of the Complaint implicates legal conclusions, no response is required. To the extent a response is required, Foras admits that there is an actual and justiciable controversy between Foras and HLK regarding the infringement of at least claims 1, 2,

3, 6, 8, 19, and 20 of the '958 Patent by HLK's Accused Products. Further, Foras admits that it has filed a complaint against Kia for patent infringement of at least claims 1, 2, 3, 6, 8, 19, and 20 of the '958 Patent. Foras lacks knowledge or information sufficient to form a belief as to the truth of the remaining allegations in paragraph 21, and therefore denies those allegations.

22. To the extent paragraph 22 of the Complaint implicates legal conclusions, no response is required. To the extent a response is required, Foras admits that the Court has personal jurisdiction over Foras for this action only, but denies any remaining allegations in paragraph 22.

23. To the extent paragraph 23 of the Complaint implicates legal conclusions, no response is required. To the extent a response is required, Foras admits that this Court is a proper venue for this action only, but denies any remaining allegations in paragraph 23.

24. To the extent paragraph 24 of the Complaint implicates legal conclusions, no response is required. To the extent a response is required, Foras admits that assignment to the Alexandria Division is proper under Local Civil Rule 3(c) for this action only, and that personal jurisdiction exists for this action only, but denies any remaining allegations in paragraph 24.

FACTUAL BACKGROUND

25. Admitted.

26. Foras admits that it asserted that Kia infringes by making, using, offering for sale, selling and/or importing "certain products and services, including without limitation the Kia K5, Telluride, and Seltos which include a Mando MRR-20 Mid-Range Radar (containing an Infineon SAK-TC297TA chipset) and/or Hella RS4 Lange Change Assist Radar (containing an Infineon SAK-TC264DA chipset) ('Accused Products'), that directly infringe, literally and/or under the doctrine of equivalents, one or more claims of the '958 Patent." Foras admits that it asserted that certain Kia products infringe at least claim 19 of the '958 Patent as described in a claim chart

attached to its complaint. Foras lacks knowledge or information sufficient to form a belief as to the truth of the remaining allegations in paragraph 26, and therefore denies those allegations.

27. Foras admits that its allegations of infringement of the '958 Patent against Kia are based on Kia's use and/or incorporation of HLK's Accused Products. Foras admits that HLK's Accused Products perform every limitation of at least claim 19 of the '958 Patent. Foras lacks knowledge or information sufficient to form a belief as to the truth of the remaining allegations in paragraph 27, and therefore denies those allegations.

28. Foras admits that it alleges that Kia "ha[s] and continue[s] to make, use, offer for sale, sell, and/or import certain products and services" "that directly infringe, literally and/or under the doctrine of equivalents, one or more claims of the '958 Patent." Foras denies the remaining allegations in paragraph 28.

29. Foras admits that, on July 21, 2023, Foras served Kia with infringement contentions contending that certain Kia automobiles, "including, but not limited to Kia K5, Telluride, Seltos, and Sorento automobiles" infringe at least claims 1, 2, 3, 6, 8, 19 and 20 of the '958 Patent. Foras denies the remaining allegations in paragraph 29.

30. Admitted.

31. Foras admits that, on November 2, 2023, it attempted to serve a subpoena for the production of documents by HLK by delivering the subpoena to a California-based subsidiary of HLK, HL Klemove America Corp and that a copy of the subpoena is attached as Exhibit 6 to HLK's Complaint. Foras denies the remaining allegations in paragraph 31.

32. Foras admits that the lawsuit against Kia remains pending. Foras lacks knowledge or information sufficient to form a belief as to the truth of the remaining allegations in paragraph 32, and therefore denies those allegations.

33. Foras lacks knowledge or information sufficient to form a belief as to the truth of the remaining allegations in paragraph 32, and therefore denies those allegations.

FIRST CLAIM FOR RELIEF

(Declaratory Judgment that Plaintiff Does Not Infringe the '958 Patent)

34. Foras repeats and incorporates by reference as if fully set forth herein its responses to paragraphs 1-33 above.

35. Foras admits that there is an actual, justiciable, substantial, and immediate controversy between Foras and HLK regarding HLK's direct and/or indirect infringement of at least claims 1, 2, 3, 6, 8, 19, and 20 of the '958 Patent. Foras denies any remaining allegations in paragraph 35.

36. Denied.

37. Denied.

DEMAND FOR A JURY TRIAL

38. This paragraph sets forth HLK's demand for a trial by jury to which no response is required.

RESERVATION OF RIGHTS

39. This paragraph sets forth HLK's reservation of rights to which no response is required.

PLAINTIFF'S PRAYER FOR RELIEF

40. This paragraph sets forth HLK's prayer for relief to which no response is required. Foras denies that HLK is entitled to any of the requested relief.

DEFENDANT'S AFFIRMATIVE DEFENSES

Without conceding that any of the following necessarily must be pleaded as an affirmative

defense, or that any of the following is not already at issue by virtue of the foregoing responses, and without prejudice to Defendant's right to plead additional defenses as discovery into the facts and circumstances of the matter warrant, Defendant asserts the following affirmative defenses.

FIRST AFFIRMATIVE DEFENSE: FAILURE TO STATE A CLAIM

41. The Complaint fails to state a claim upon which relief can be granted.

SECOND AFFIRMATIVE DEFENSE: INFRINGEMENT

42. HLK's claim for declaratory judgment of non-infringement of U.S. Patent No. 7,502,958 fails because HLK infringes the '958 Patent directly and indirectly in violation of 35 U.S.C. § 271(a)-(c). Foras' infringement allegations are detailed below in Counterclaim I.

THIRD AFFIRMATIVE DEFENSE: EQUITABLE DEFENSES

43. Each of HLK's claims is barred, in whole or in part, by the equitable defenses of estoppel, waiver, laches, and/or unclean hands.

FOURTH AFFIRMATIVE DEFENSE: FAILURE TO STATE AN EXCEPTIONAL CASE

44. The Complaint fails to state a claim for an exceptional case under 35 U.S.C. § 285.

DEFENDANT'S COUNTERCLAIMS

Foras asserts the following counterclaims against HLK:

THE PARTIES

1. Defendant Foras Technologies Limited ("Foras" or "Defendant") is a private company limited by shares organized and existing under the law of Ireland, with its principal place of business at The Hyde Building, Suite 23, The Park, Carrickmines, Dublin 18, Ireland. Foras is the sole owner by assignment of all right, title, and interest in United States Patent No. 7,502,958

(“the ’958 Patent”) (the “Asserted Patent”), including the right to recover for past, present, and future infringement.

2. Plaintiff HL Klemove Corp. (“HLK” or “Plaintiff”) is a corporation organized under the laws of the Republic of Korea with its principal place of business at 224 Harmony-ro, Yeonsu-gu, Incheon, Republic of Korea.

3. On information and belief, HLK is responsible for importing, making, marketing, distributing, offering for sale, and/or selling Advanced Driver Assistance Systems (ADAS), such as the Mando MRR-20 Mid-Range Radar (containing an Infineon SAK-TC297TA chipset) and/or Hella RS4 Lane Change Assist Radar (containing an Infineon SAK-TC264DA chipset) (“Accused Products”), throughout the United States, including this District. On information and belief, Kia Corp. and/or Kia America, Inc. (“Kia”) is one of HLK’s customers, as well as other major automobile manufacturers. On information and belief, HLK offers for sale and/or sells the Accused Products to Kia and other automobile manufacturers in the United States.

JURISDICTION AND VENUE

4. This action arises under the patent laws of the United States, Title 35 of the United States Code. This Court has original and exclusive subject matter jurisdiction pursuant to 28 U.S.C. §§ 1331 and 1338(a).

5. This Court has personal jurisdiction over HLK because HLK submitted themselves to this Court’s jurisdiction by initiating this lawsuit.

6. Venue is proper for Foras’ counterclaims under 28 U.S.C. § 1391.

FIRST COUNTERCLAIM **(Infringement of U.S. Patent No. 7,502,958)**

7. Foras realleges and incorporates by reference the foregoing paragraphs as if fully set forth herein.

8. Foras owns by assignment all rights, title, and interest, including the right to recover damages for past, present, and future infringement, in U.S. Patent No. 7,502,958, entitled “System and method for providing firmware recoverable lockstep protection.” The ’958 Patent was duly and legally issued by the United States Patent and Trademark Office on March 10, 2009. A true and correct copy of the ’958 Patent is attached as **Exhibit A**.

9. On information and belief, HLK has and continues to make, use, offer for sale, sell, and/or import certain products and services, including Advanced Driver Assistance Systems (ADAS), such as the Mando MRR-20 Mid-Range Radar (containing an Infineon SAK-TC297TA chipset) and/or Hella RS4 Lange Change Assist Radar (containing an Infineon SAK-TC264DA chipset) products (“Accused Products”), that directly infringe, literally and/or under the doctrine of equivalents, one or more claims of the ’958 Patent. Identification of the Accused Products will be provided in Foras’ infringement contentions disclosed pursuant to the Court’s scheduling order.

10. The Accused Products satisfy all claim limitations of one or more claims of the ’958 Patent. Claim charts comparing exemplary independent claim 19 of the ’958 Patent to representative Accused Products are attached as **Exhibit B**.

11. By making, using, offering for sale, selling and/or importing into the United States the Accused Products, HLK has injured Foras and is liable for infringement of the ’958 Patent pursuant to 35 U.S.C. § 271(a).

12. HLK also knowingly and intentionally induces infringement of one or more claims of the ’958 Patent in violation of 35 U.S.C. § 271(b). At least as of the filing and service of its Complaint, HLK has knowledge of the ’958 Patent and the infringing nature of the Accused Products through, for example, the ’958 Patent claim chart from the Kia case that was attached to HLK’s Complaint as Exhibit 3. Despite this knowledge of the ’958 Patent, HLK continues to

actively encourage and instruct its customers to use the Accused Products in ways that directly infringe the '958 Patent. HLK engages in many activities that encourage its customers to infringe the '958 Patent, including (i) advertising and promotion efforts for the Accused Products; (ii) publication of demonstrational videos on its website; (iii) publication of advertising and marketing materials that purport to describe the alleged benefits of the Accused Products; and (iv) publication of user manuals that instruct customers on how to install and use the Accused Products in an infringing manner. For example, HLK provides its customers, such as Kia, with user guides for the Mando MRR-20 Mid-Range Radar, which includes instructions on how to use and operate the product. See **Exhibit C** (MRR-20 User Guide) at pp. 9-12. Similarly, HLK provides its customers, such as Kia, with a user guide for the Hella RS4 Lange Change Assist Radar, which includes instructions on how to use and operate the product. See **Exhibit D** (RS 5.4 User Guide) at pp. 2-7.

13. HLK further publishes marketing materials characterizing its radar products as “key” products and touting their success:

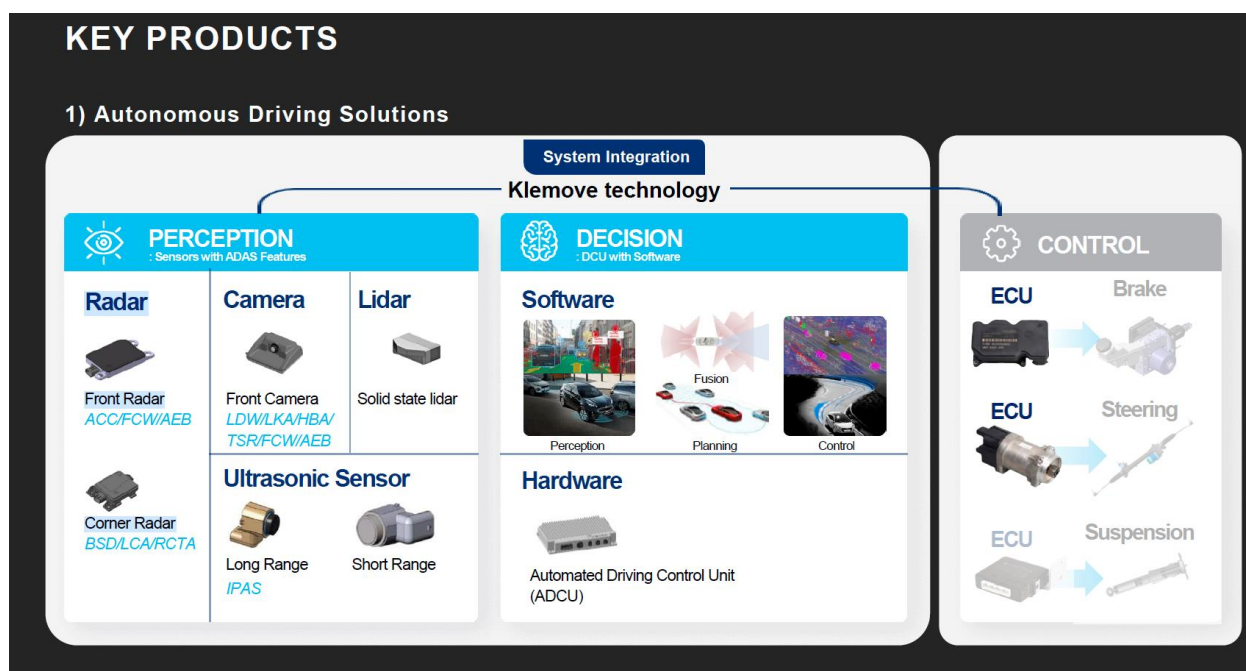


Exhibit E (HL Klemove Marketing Presentation) at p. 16.

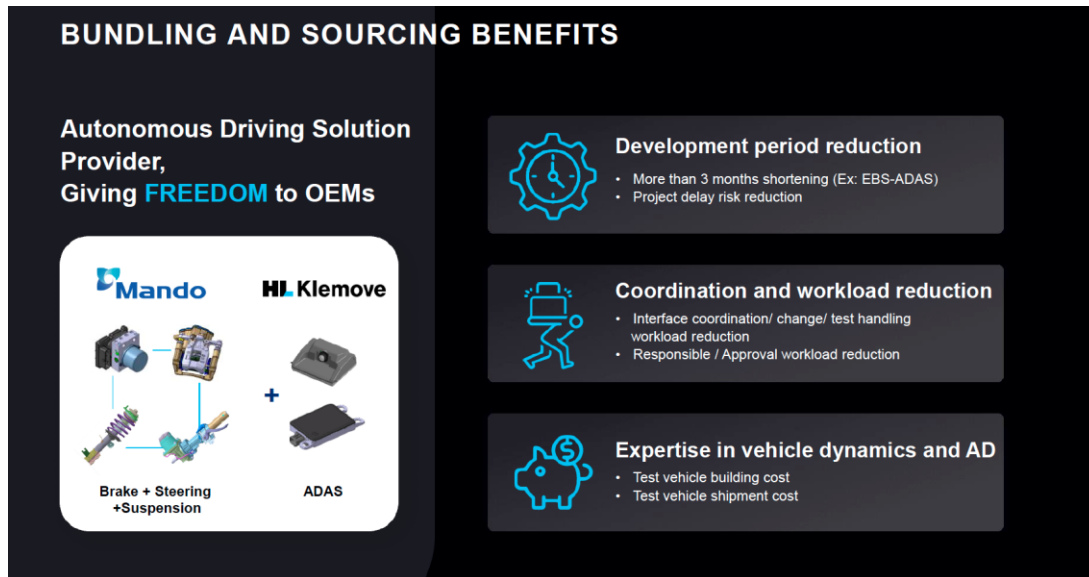


Exhibit E (HL Klemove Marketing Presentation) at p. 19.

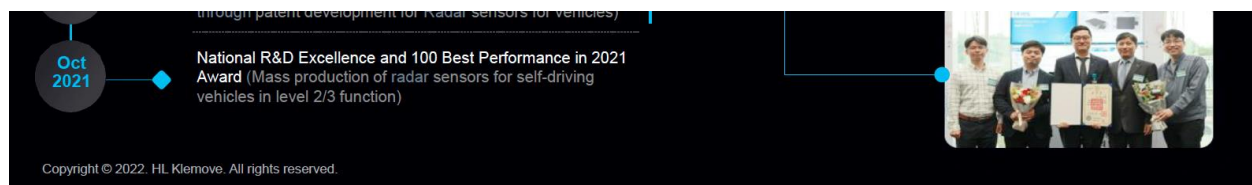


Exhibit E (HL Klemove Marketing Presentation) at p. 20.

14. HLK provides these instructions, user manuals, and other materials knowing and intending (or with willful blindness to the fact) that its customers, such as Kia and other automobile manufacturers, will commit these infringing acts. HLK also continues to make, use, offer for sale, sell, and/or import the Accused Products, despite its knowledge of the '958 Patent, thereby specifically intending for and inducing its customers to infringe the '958 Patent through the customers' normal and customary use of the Accused Products.

15. HLK has also infringed, and continues to infringe, one or more claims of the '958 Patent by selling, offering for sale, or importing into the United States, the Accused Products, knowing that the Accused Products constitute a material part of the inventions claimed in the '958

Patent, are especially make or adapted to infringe the '958 Patent, and are not staple articles or commodities of commerce suitable for non-infringing use. At least as of the filing and service of HLK's Complaint, HLK has had knowledge of the '958 Patent and the infringing nature of the Accused Products through, for example, the '958 Patent claim chart from the Kia case served with HLK's Complaint as Exhibit 3. HLK has been, and currently is, contributorily infringing the '958 Patent in violation of 35 U.S.C. §§ 271(c) and/or (f). For example, HLK's Accused Products constitute a material part of the inventions claimed in the '958 Patent, are especially made or adapted to infringe the '958 Patent, and are not staple articles or commodities of commerce suitable for non-infringing use, as demonstrated by the evidence in **Exhibit B**.

16. On information and belief, Foras (including its predecessors and any licensees) complied with 35 U.S.C. §287 at least prior to June 30, 2022 because Foras, its predecessors, and any licensees did not make, offer for sale, or sell products that practice(d) the '958 Patent during the relevant time period or were not otherwise required to mark during the relevant time period.

17. As a result of HLK's direct infringement of the '958 Patent, Foras is entitled to monetary damages (past, present, and future) in an amount adequate to compensate for HLK's infringement, but in no event less than a reasonable royalty for the use made of the invention by HLK, together with interest and costs as fixed by the Court.

18. As a result of HLK's indirect infringement of the '958 Patent, Foras is entitled to monetary damages (present and future) in an amount adequate to compensate for HLK's infringement, but in no event less than a reasonable royalty for the use made of the invention by HLK, together with interest and costs as fixed by the Court, accruing as of the time HLK obtained knowledge of the '958 Patent.

PRAYER FOR RELIEF

WHEREFORE, Foras respectfully requests that this Court enter:

- a. A judgment in favor of Foras that HLK has infringed, either literally and/or under the doctrine of equivalents, the '958 Patent;
- b. A judgment and order requiring HLK to pay Foras its damages (past, present, and future), costs, expenses, and pre-judgment and post-judgment interest for their infringement of the '958 Patent;
- c. A judgment and order requiring HLK to pay Foras compulsory ongoing licensing fees, as determined by the Court;
- d. A judgment and order requiring HLK to provide an accounting and to pay supplemental damages to Foras, including without limitation, pre-judgment and post-judgment interest and compensation for infringing products released after the filing of this case that are not colorably different from the Accused Products;
- e. A judgment and order finding that this is an exceptional case within the meaning of 35 U.S.C. § 285 and awarding to Foras its reasonable attorneys' fees against HLK; and
- f. Any and all other relief as the Court may deem appropriate and just under the circumstances.

DEMAND FOR JURY TRIAL

Plaintiff, under Rule 38 of the Federal Rules of Civil Procedure, requests a trial by jury of any issues so triable by right.

Dated: February 12, 2024

Respectfully submitted,

/s/ Robert E. Scully, Jr.

Robert E. Scully, Jr. (VSB No. 19218)

Michael K. Kim (VSB No. 82922)

BLANKINGSHIP & KEITH, P.C.

4020 University Drive, Suite 300

Fairfax, VA 22030

Tel.: (703) 691-1235

Fax: (703) 691-3913

rscully@bklawva.com

Brett E. Cooper (NY SBN 4011011)

(pro hac vice motion forthcoming)

BC LAW GROUP, P.C.

200 Madison Avenue, 24th Floor

New York, NY 10016

Tel.: (212) 951-0100

Fax: (646) 293-2201

bcooper@bclgpc.com

*Attorneys for Defendant/Counterclaimant
Foras Technologies Limited*

CERTIFICATE OF SERVICE

I hereby certify that on February 12, 2024, I caused a copy of the foregoing to be electronically filed using the CM/ECF system, which will send notification to counsel of record of such filing by operation of the Court's electronic system. Parties may access this filing via the Court's electronic system.

/s/ Robert E. Scully, Jr.

Robert E. Scully, Jr. (VSB No. 19218)

EXHIBIT A

US007502958B2

(12) **United States Patent**
Michaelis et al.

(10) **Patent No.:** **US 7,502,958 B2**
(45) **Date of Patent:** ***Mar. 10, 2009**

(54) **SYSTEM AND METHOD FOR PROVIDING
FIRMWARE RECOVERABLE LOCKSTEP
PROTECTION**

(75) Inventors: **Scott L. Michaelis**, Plano, TX (US);
Anurupa Rajkumari, Round Rock, TX
(US); **William B. McHardy**, Fort
Collins, CO (US)

(73) Assignee: **Hewlett-Packard Development
Company, L.P.**, Houston, TX (US)

(*) Notice: Subject to any disclaimer, the term of this
patent is extended or adjusted under 35
U.S.C. 154(b) by 626 days.

This patent is subject to a terminal dis-
claimer.

5,915,082 A *	6/1999	Marshall et al.	714/11
5,991,900 A	11/1999	Garnett	
6,065,135 A	5/2000	Marshall et al.	
6,141,770 A	10/2000	Fuchs et al.	
6,148,348 A	11/2000	Garnett et al.	
6,220,581 B1	4/2001	Mueller	
6,263,452 B1	7/2001	Jewett et al.	
6,438,687 B2	8/2002	Klein	
6,473,869 B2	10/2002	Bissett et al.	
6,560,682 B1	5/2003	Miller et al.	
6,604,177 B1	8/2003	Kondo et al.	
6,615,366 B1	9/2003	Grochowski et al.	
6,625,749 B1 *	9/2003	Quach	714/10
6,675,324 B2	1/2004	Marisetty et al.	
6,687,851 B1	2/2004	Somers et al.	
6,754,787 B2	6/2004	Miller et al.	
6,920,581 B2 *	7/2005	Bigbee et al.	714/10

(21) Appl. No.: **10/973,076**

(22) Filed: **Oct. 25, 2004**

(65) **Prior Publication Data**

US 2006/0107107 A1 May 18, 2006

(51) **Int. Cl.**
G06F 11/00 (2006.01)

(52) **U.S. Cl.** **714/12; 712/31**

(58) **Field of Classification Search** 714/10-13,
714/712, 31, 34; 712/31

See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

4,358,823 A	11/1982	McDonald et al.
5,226,152 A	7/1993	Klug et al.
5,249,188 A	9/1993	McDonald
5,537,655 A	7/1996	Truong
5,675,794 A	10/1997	Meredith
5,751,932 A	5/1998	Horst et al.
5,758,058 A	5/1998	Milburn
5,764,660 A	6/1998	Mohat
5,896,523 A *	4/1999	Bissett et al.

(Continued)

OTHER PUBLICATIONS

Filed concurrently herewith, entitled "System and Method for Using
Information Relating to a Detected Loss of Lockstep for Determining
a Responsive Action".

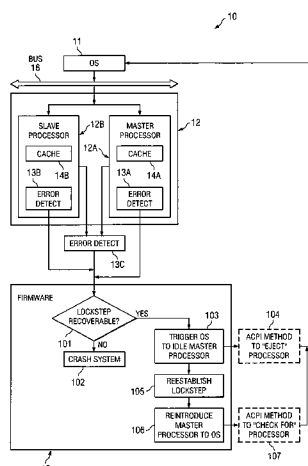
(Continued)

Primary Examiner—Dieu-Minh Le

(57) **ABSTRACT**

According to at least one embodiment, a method comprises
detecting loss of lockstep for a pair of processors. The method
further comprises triggering, by firmware, an operating sys-
tem to idle the processors, and recovering, by the firmware,
lockstep between the pair of processors. After lockstep is
recovered between the pair of processors, the method further
comprises triggering, by the firmware, the operating system
to recognize the processors as being available for receiving
instructions.

25 Claims, 5 Drawing Sheets



US 7,502,958 B2

Page 2

U.S. PATENT DOCUMENTS

6,948,092 B2 9/2005 Kondo et al.
 7,003,691 B2 2/2006 Safford et al.
 7,020,800 B2 * 3/2006 Fu et al. 714/38
 7,085,959 B2 * 8/2006 Safford 714/11
 7,134,047 B2 11/2006 Quach
 7,155,721 B2 12/2006 Safford et al.
 7,191,292 B2 * 3/2007 Chaudhry et al. 711/133
 7,225,355 B2 5/2007 Yamazaki et al.
 7,237,144 B2 6/2007 Safford et al.
 7,251,746 B2 7/2007 Fox et al.
 2002/0144177 A1 10/2002 Kondo et al.
 2002/0152420 A1 10/2002 Chaudhry et al.
 2003/0051190 A1 3/2003 Marisetty et al.
 2003/0070050 A1 4/2003 Miller et al.
 2003/0126498 A1 7/2003 Bigbee et al.
 2003/0135711 A1 7/2003 Shoemaker et al.
 2003/0140281 A1 * 7/2003 Fu et al. 714/38
 2004/0006722 A1 1/2004 Safford
 2004/0019771 A1 1/2004 Quach
 2004/0078650 A1 4/2004 Safford et al.
 2004/0078651 A1 4/2004 Stafford et al.
 2004/0153857 A1 8/2004 Yamazaki et al.
 2005/0172164 A1 * 8/2005 Fox et al. 714/13
 2005/0240806 A1 * 10/2005 Bruckert et al. 714/6
 2005/0240811 A1 10/2005 Stafford et al.
 2005/0240829 A1 * 10/2005 Safford et al. 714/43
 2006/0085677 A1 4/2006 Stafford et al.
 2006/0136672 A1 * 6/2006 Chaudhry et al. 711/122
 2006/0248384 A1 11/2006 Stafford

2006/0248684 A1 11/2006 Vosbikian et al.

OTHER PUBLICATIONS

Filed concurrently herewith, entitled "System and Method for Establishing a Spare Processor for Recovering From Loss of Lockstep in a Boot Processor".

Filed concurrently herewith, entitled "System and Method for Configuring Lockstep Mode of a Processor Module".

Filed concurrently herewith, entitled System and Method for Maintaining in a Multi-Processor System a Spare Processor That is in Lockstep for Use in Recovering From Loss of Lockstep for Another Processor.

Filed concurrently herewith, entitled "System and Method for Switching the Role of Boot Processor to a Spare Processor Responsive to Detection of Loss of Lockstep in a Boot Processor".

Filed concurrently herewith, entitled "System and Method for Reestablishing Lockstep for a Processor Module for Which Loss of Lockstep Is Detected".

Filed concurrently herewith, entitled "System and Method for System Firmware Causing an Operating System to Idle a Processor".

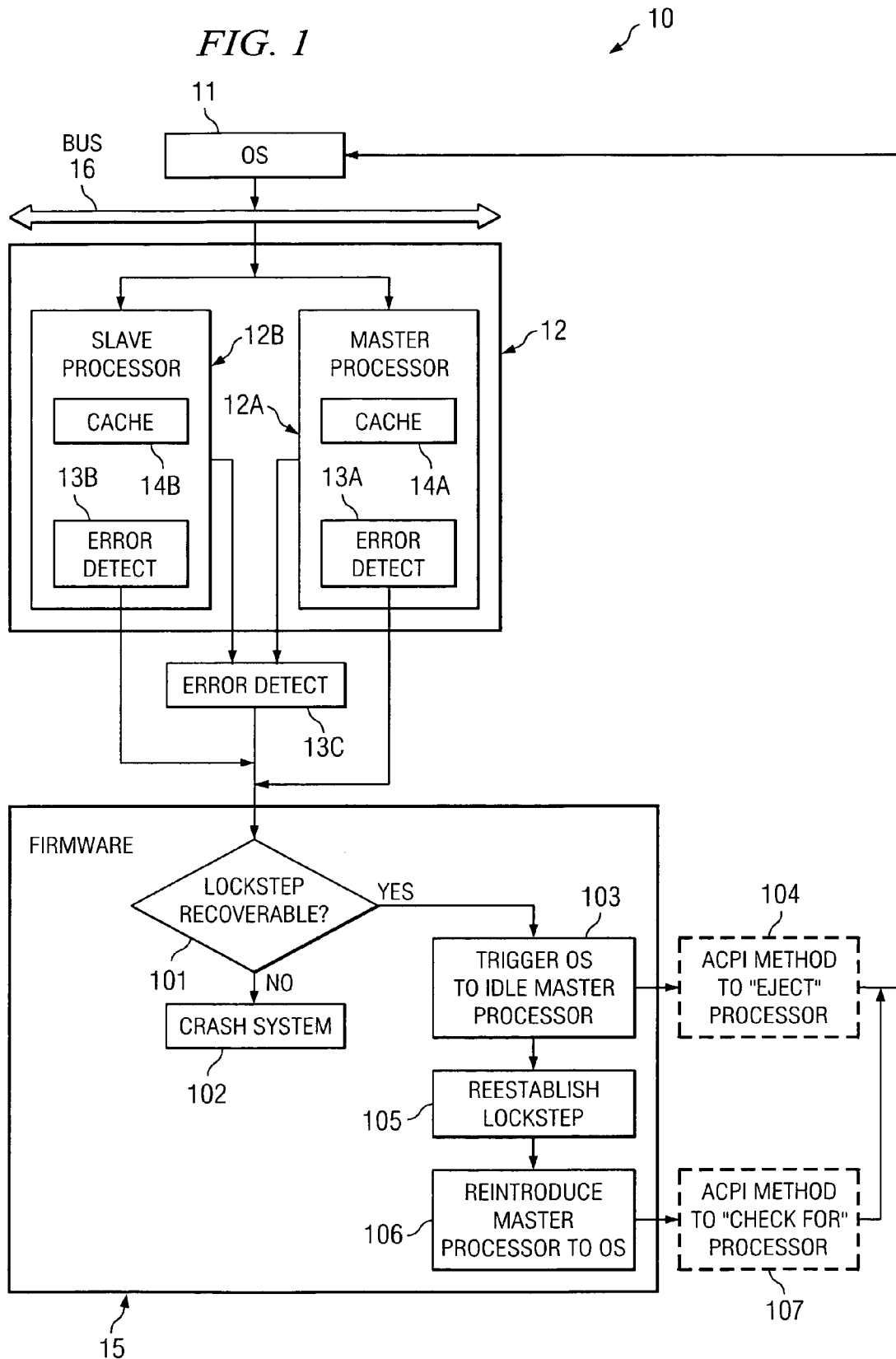
Filed concurrently herewith, entitled "System and Method for Reintroducing a Processor Module to an Operating System After Lockstep Recovery".

Filed concurrently herewith, entitled "System and Method For System Firmware Causing an Operating System to Idle a Processor".

Filed concurrently herewith, entitled "System and Method For Reintroducing a Processor Module to an Operating System After Lockstep Recovery".

* cited by examiner

FIG. 1



U.S. Patent

Mar. 10, 2009

Sheet 2 of 5

US 7,502,958 B2

FIG. 2

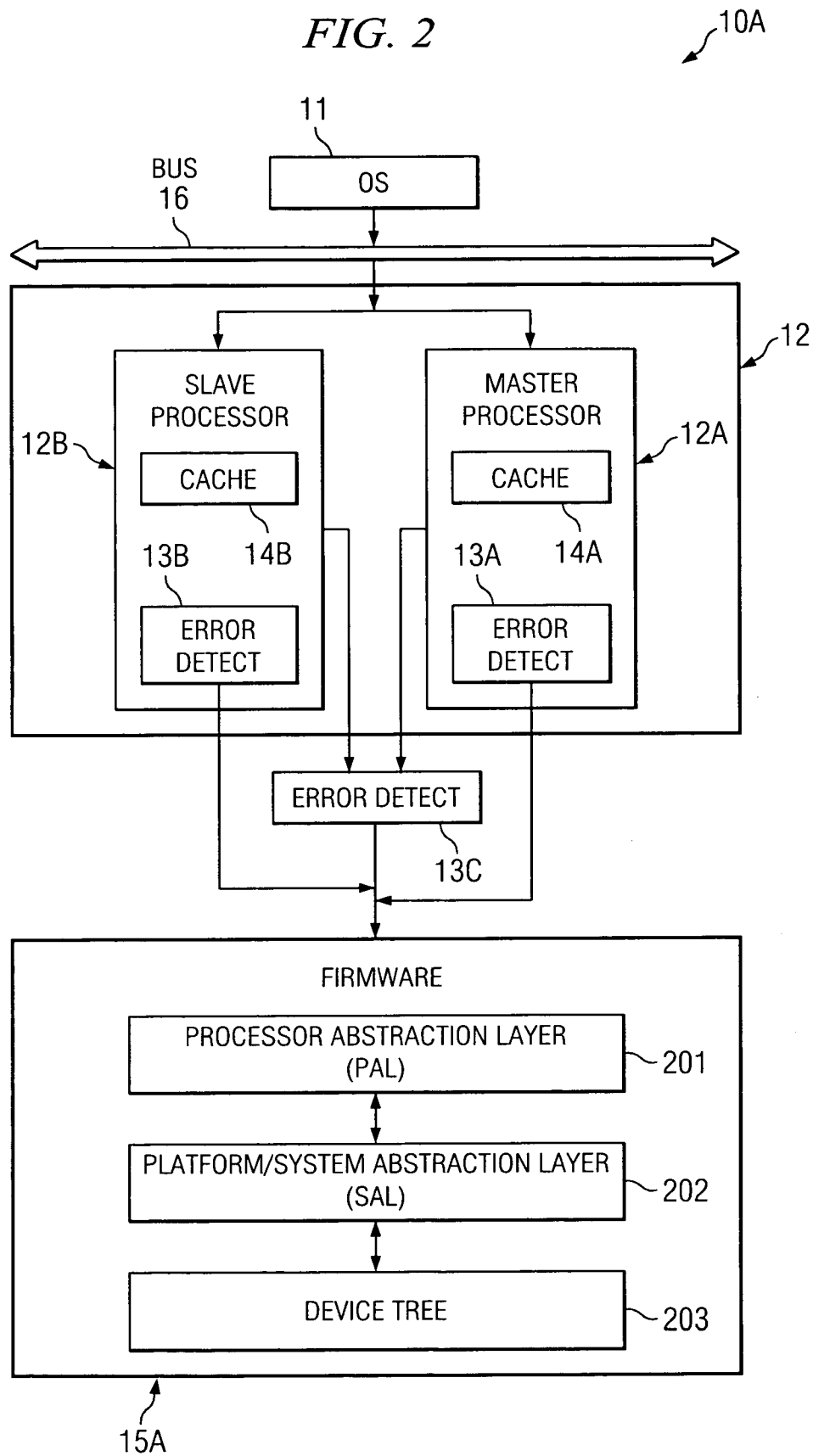


FIG. 3

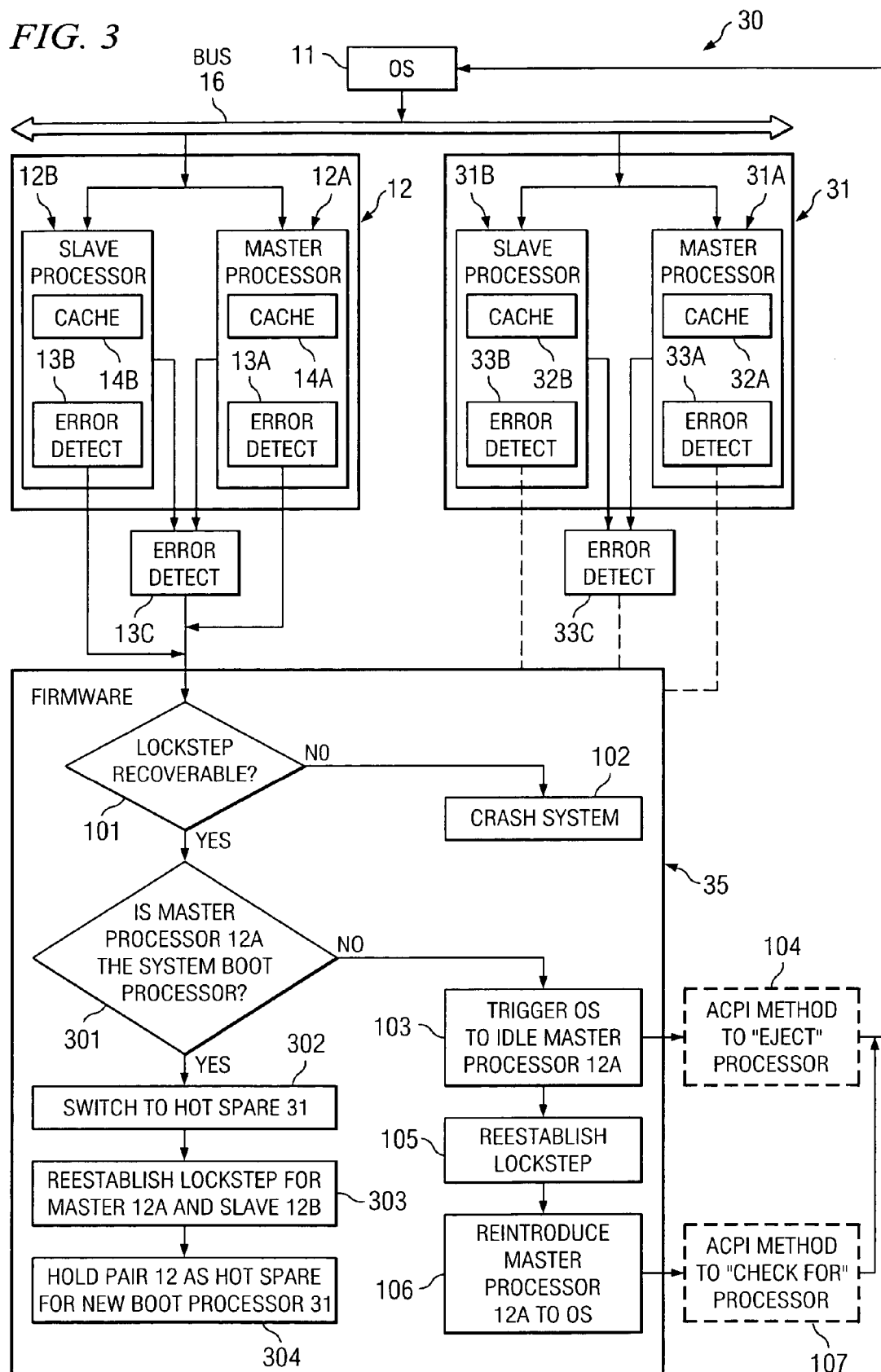


FIG. 4

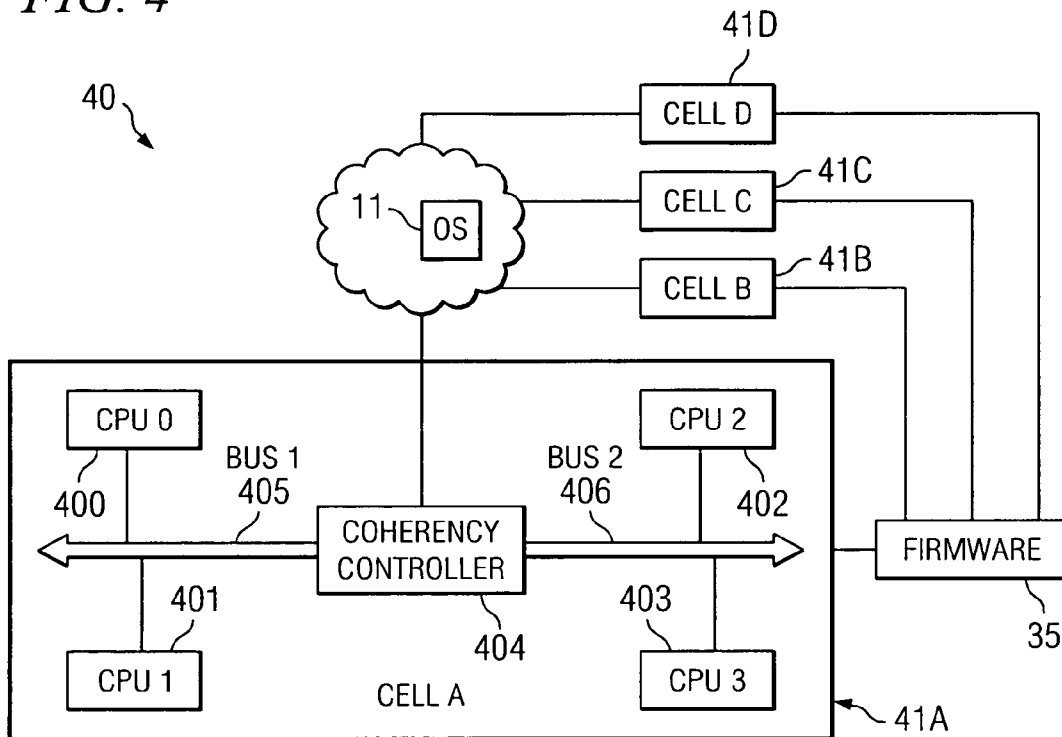


FIG. 6

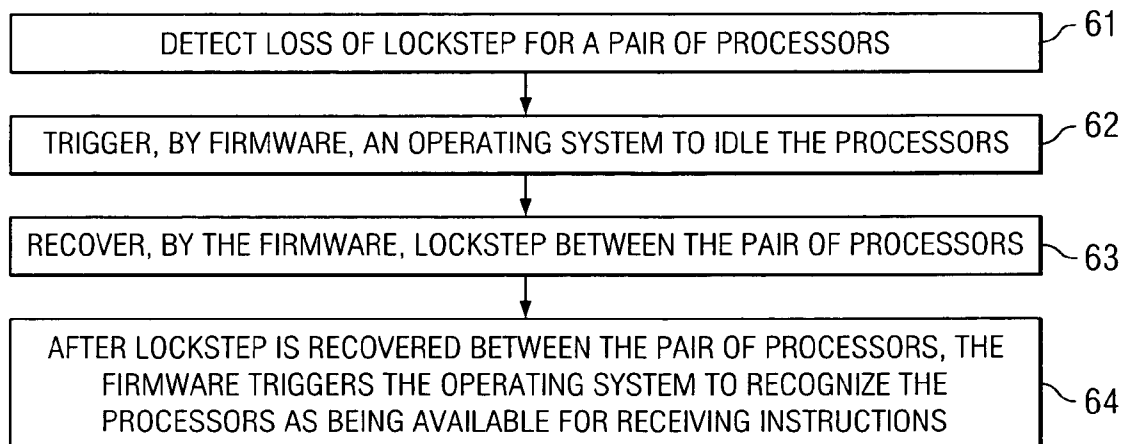
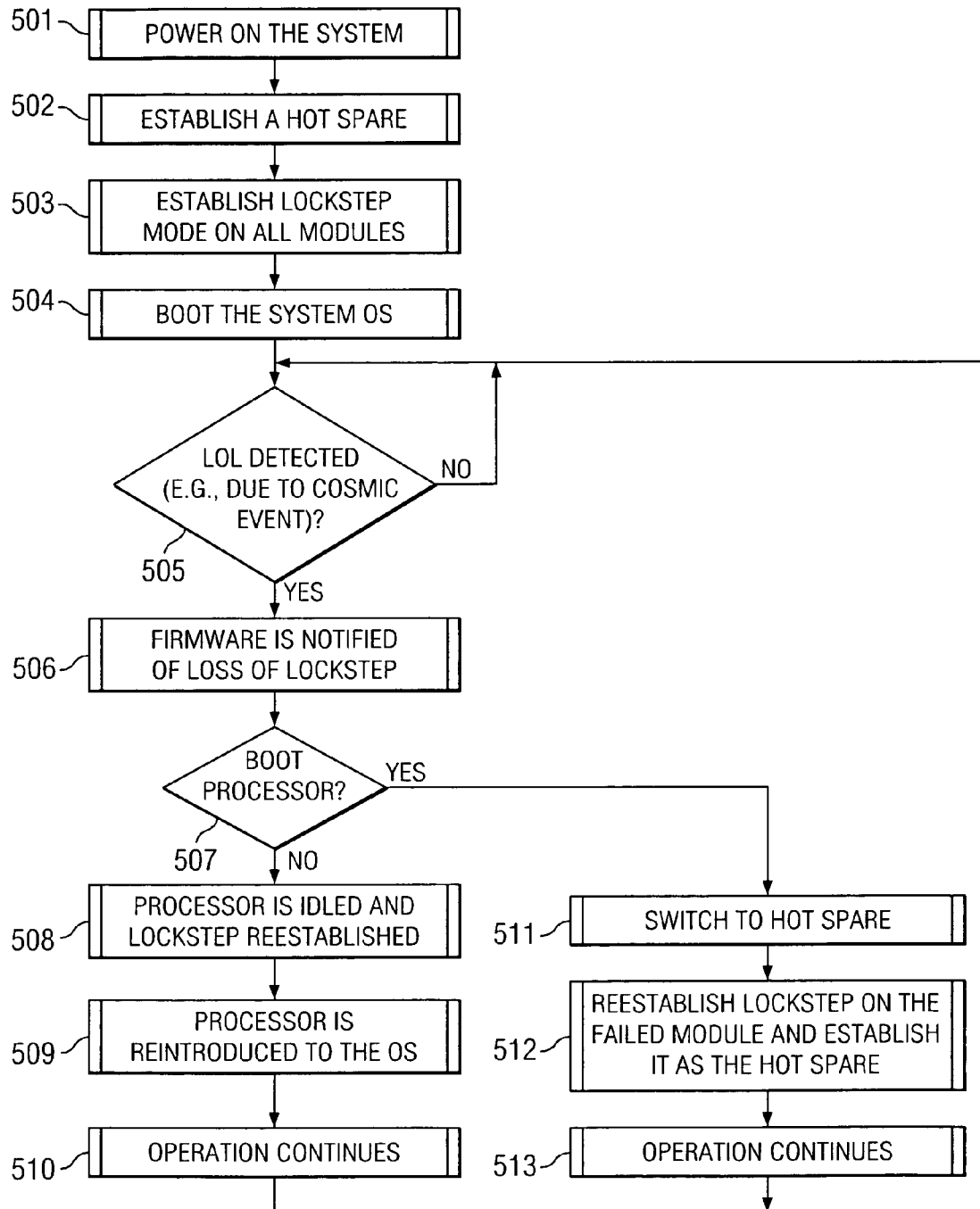


FIG. 5

US 7,502,958 B2

1

SYSTEM AND METHOD FOR PROVIDING FIRMWARE RECOVERABLE LOCKSTEP PROTECTION

CROSS-REFERENCE TO RELATED APPLICATIONS

The present application is related to the following concurrently filed and commonly assigned U.S. patent applications: Ser. No. 10/973,077 titled "SYSTEM AND METHOD FOR ESTABLISHING A SPARE PROCESSOR FOR RECOVERING FROM LOSS OF LOCKSTEP IN A BOOT PROCESSOR"; Ser. No. 10/973,004 titled "SYSTEM AND METHOD FOR CONFIGURING LOCKSTEP MODE OF A PROCESSOR MODULE"; Ser. No. 10/972,835 titled "SYSTEM AND METHOD FOR USING INFORMATION RELATING TO A DETECTED LOSS OF LOCKSTEP FOR DETERMINING A RESPONSIVE ACTION"; Ser. No. 10/972,588 titled "SYSTEM AND METHOD FOR SWITCHING THE ROLE OF BOOT PROCESSOR TO A SPARE PROCESSOR RESPONSIVE TO DETECTION OF LOSS OF LOCKSTEP IN A BOOT PROCESSOR"; Ser. No. 10/973,003 titled "SYSTEM AND METHOD FOR REESTABLISHING LOCKSTEP FOR A PROCESSOR MODULE FOR WHICH LOSS OF LOCKSTEP IS DETECTED"; Ser. No. 10/972,888 titled "SYSTEM AND METHOD FOR SYSTEM FIRMWARE CAUSING AN OPERATING SYSTEM TO IDLE A PROCESSOR"; Ser. No. 10/973,075 titled "SYSTEM AND METHOD FOR REINTRODUCING A PROCESSOR MODULE TO AN OPERATING SYSTEM AFTER LOCKSTEP RECOVERY"; and Ser. No. 10/972,796 titled "SYSTEM AND METHOD FOR MAINTAINING IN A MULTI-PROCESSOR SYSTEM A SPARE PROCESSOR THAT IS IN LOCKSTEP FOR USE IN RECOVERING FROM LOSS OF LOCKSTEP FOR ANOTHER PROCESSOR", the disclosures of which are hereby incorporated herein by reference.

DESCRIPTION OF RELATED ART

Silent Data Corruption ("SDC") is a difficult problem in the computing industry. In general, SDC refers to data that is corrupt, but which the system does not detect as being corrupt. SDCs primarily occur due to one of two factors: a) a broken hardware unit or b) a "cosmic" event that causes values to change somewhere in the system. Broken hardware means that a "trusted" piece of hardware is silently giving wrong answers. For example, the arithmetic unit in a processor is instructed to add 1+1 and it returns the incorrect answer 3 instead of the correct answer 2. An example of a cosmic event is when a charged particle (e.g., alpha particle or cosmic ray) strikes a region of a computing system and causes some bits to change value (e.g., from a 0 to a 1 or from a 1 to a 0).

Numerous techniques have been developed for detecting SDC to prevent the SDC from remaining "silent" or "undetected" within a system, as well as preventing such SDC from propagating through the system. Examples of these techniques include parity-based mechanisms and error correcting codes (ECCs) on buses and memory locations, as well as checksums and/or cyclic redundancy checks (CRC) over regions of memory. Parity-based mechanisms are often employed in processors, wherein a parity bit is associated with each block of data when it is stored. The parity bit is set to one or zero according to whether there is an odd or even number of ones in the data block. When the data block is read out of its storage location, the number of ones in the block is compared with the parity bit. A discrepancy between the

2

values indicates that the data block has been corrupted. ECCs are parity-based mechanisms that track additional information for each data block. The additional information allows the corrupted bit(s) to be identified and corrected.

Parity/ECC mechanisms have been employed extensively for caches, memories, and similar data storage arrays. In the remaining circuitry on a processor, such as data paths, control logic, execution logic, and registers (the "execution core"), it is more difficult to apply parity/ECC mechanisms for SDC detection. Thus, there is typically some unprotected area on a processor in which data corruption may occur and the parity/ECC mechanisms do not prevent the corrupted data from actually making it out onto the system bus. One approach to SDC detection in an execution core (or other unprotected area of the processor chip) is to employ "lockstep processing." Generally, in lockstep processing two processors are paired together, and the two processors perform exactly the same operations and the results are compared (e.g., with an XOR gate). If there is ever a discrepancy between the results of the lockstep processors, an error is signaled. The odds of two processors experiencing the exact same error at the exact same moment (e.g., due to a cosmic event occurring in both processors at exactly the same time or due to a mechanical failure occurring in each processor at exactly the same time) is nearly zero.

A pair of lockstep processors may, from time to time, lose their lockstep. "Loss of lockstep" (or "LOL") is used broadly herein to refer to any error in the pair of lockstep processors. One example of LOL is detection of data corruption (e.g., data cache error) in one of the processors by a parity-based mechanism and/or ECC mechanism. Another example of LOL is detection of the output of the paired processors not matching, which is referred to herein as a "lockstep mismatch." It should be recognized that in some cases the data in the cache of a processor may become corrupt (e.g., due to a cosmic event), which once detected (e.g., by a parity-based mechanism or ECC mechanism of the processor) results in LOL. Of course, unless such corrupt data is acted upon by the processor, the output of that processor will not fail to match the output of its paired processor and thus a "lockstep mismatch" will not occur. For example, suppose that a value of "1" is stored to first location of cache in each of a pair of lockstep processors and a value of "1" is also stored to a second location of cache in each of the pair of lockstep processors. Further suppose that a cosmic event occurs for a first one of the processors, resulting in the first location of its cache being changed from "1" to "0", and thus corrupted. This data corruption in the first processor is a LOL for the pair. An error detection mechanism of this first processor may detect the data corruption, thus detecting the LOL. If the processors are instructed to act on the data of their first cache locations, then a lockstep mismatch will occur as the output of each of the processors will not match. For instance, if the processors each add the data stored to the first location of their respective cache with the data stored to the second location of their respective cache, the first processor (having the corrupt data) will output a result of "1" ($0+1=1$) while the second processor outputs a result of "2" ($1+1=2$), and thus their respective outputs will not match.

By employing such techniques as parity-based error detection mechanisms and output comparisons for lockstep paired processors, SDC detection can be enhanced such that practically no SDC occurring in a processor goes undetected (and thus such SDC does not remain "silent") but instead results in detection of LOL. However, the issue then becomes how best for the system to respond to detected LOL. The traditional response to detected LOL has been to crash the system to

US 7,502,958 B2

3

ensure that the detected error is not propagated through the system. That is, LOL in one pair of lockstep processors in a system halts processing of the system even if other processors that have not encountered an error are present in the system. However, with the increased desire for many systems to maintain high availability, crashing the system each time LOL is detected is not an attractive proposition. This is particularly unattractive for large systems having many processors because cosmic events typically occur more frequently as the processor count goes up, which would result in much more frequent system crashes in those large systems. High availability is a major desire for many customers having large, multi-processor systems, and thus having their system crash every few weeks is not an attractive option. Of course, permitting corrupt data to propagate through the system is also not a viable option.

Prior solutions attempting to resolve at least some detected SDCs without requiring the system to be crashed have been Operating System ("OS") centric. That is, in certain solutions the OS has been implemented in a manner to recover from a detected LOL without necessarily crashing the system. This OS-centric type of solution requires a lot of processor and platform specific knowledge to be embedded in the OS, and thus requires that the OS provider maintain the OS up-to-date as changes occur in later versions of the processors and platforms in which the OS is to be used. This is such a large burden that most commonly used OSs do not support lockstep recovery.

Certain solutions have attempted to recover from a LOL without involving the OS in such recovery procedure. For instance, in one technique upon LOL being detected, firmware is used to save the state of one of the processors in a lockstep pair (the processor that is considered "good") to memory, and then both processors of the pair are reset and reinitialized. Thereafter, the state is copied from the memory to each of the processors in the lockstep pair. This technique makes the processors unavailable for an amount of time without the OS having any knowledge regarding this unavailability, and if the amount of time required for recovery is too long, the system may crash. That is, typically, if a processor is unresponsive for X amount of time, the OS will assume that the processor is hung and will crashdump the system so that the problem can be diagnosed. Further, in the event that a processor in the pair cannot be reset and reinitialized (e.g., the processor has a physical problem and fails to pass its self-test), this technique results in crashing the system.

BRIEF SUMMARY OF THE INVENTION

According to at least one embodiment, a method comprises detecting loss of lockstep for a pair of processors. The method further comprises triggering, by firmware, an operating system to idle the processors, and recovering, by the firmware, lockstep between the pair of processors. After lockstep is recovered between the pair of processors, the method further comprises triggering, by the firmware, the operating system to recognize the processors as being available for receiving instructions.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. 1 shows an example embodiment of a system that uses firmware for controlling recovery from detected loss of lockstep (LOL);

FIG. 2 shows a block diagram of one embodiment implemented for the IA-64 processor architecture;

4

FIG. 3 shows an example system in which firmware utilizes a hot spare processor for recovering from LOL for the system's boot processor;

FIG. 4 shows an example system having multi-processor cells in which an embodiment for using firmware for recovering from LOL may be employed;

FIG. 5 shows a detailed operational flow diagram for one embodiment; and

FIG. 6 shows a more general operational flow diagram for certain embodiments.

DETAILED DESCRIPTION

Turning to FIG. 1, an example embodiment of a system 10 that uses firmware for controlling recovery from detected loss of lockstep (LOL) is shown. System 10 includes OS 11, as well as master processor 12A and slave processor 12B (collectively referred to as a lockstep processor pair 12). In certain implementations the lockstep processor pair 12 may be implemented on a single silicon chip, which is referred to as a "dual core processor" in which master processor 12A is a first core and slave processor 12B is a second core. Master processor 12A includes cache 14A, and slave processor 12B includes cache 14B. OS 11 and lockstep processor pair 12 are communicatively coupled to bus 16. Typically, master processor 12A and slave processor 12B are coupled to bus 16 via an interface that allows each of such processors to receive the same instructions to process, but such interface only communicates the output of master processor 12A back onto bus 16. The output of slave processor 12B is used solely for checking the output of master processor 12A. While only one lockstep processor pair 12 is shown for simplicity in the example of FIG. 1, system 10 may include any number of such lockstep processor pairs. As one specific example, system 10 may have 64 lockstep processor pairs, wherein the master processors of the pairs may perform parallel processing for the system.

In this example, master processor 12A includes error detect logic 13A, and slave processor 12B includes error detect logic 13B. While shown as included in each of the processors 12A and 12B in this example, in certain embodiments the error detect logic 13A and 13B may be implemented external to processors 12A and 12B. Error detect logic 13A and 13B include logic for detecting errors, such as data cache errors, present in their respective processors 12A and 12B. Examples of error detect logic 13A and 13B include known parity-based mechanisms and ECC mechanisms. Error detect logic 13C is also included, which may include an XOR (exclusive OR) gate, for detecting a lockstep mismatch between master processor 12A and slave processor 12B. As mentioned above, a lockstep mismatch refers to the output of master processor 12A and slave processor 12B failing to match. While shown as external to the lockstep processor pair 12 in this example, in certain embodiments error detect logic 13C may be implemented on a common silicon chip with processors 12A and 12B.

Lockstep mismatch is one way of detecting a LOL between the master processor 12A and slave processor 12B. A detection of an error by either of error detect logic 13A and 13B also provides detection of LOL in the processors 12A and 12B. Because the detection of LOL by error detect logic 13A and 13B may occur before an actual lockstep mismatch occurs, the detection of LOL by error detect logic 13A and 13B may be referred to as a detection of a "precursor to lockstep mismatch". In other words, once an error (e.g., corrupt data) is detected by error detect logic 13A or 13B, such error may eventually propagate to a lockstep mismatch error that is detectable by error detect logic 13C.

US 7,502,958 B2

5

Firmware 15 is also included in system 10, which in this embodiment is invoked upon an error being detected by any of the error detect logics 13A, 13B, and 13C. In certain embodiments, processors 12A and 12B are processors from the Itanium Processor Family (IPF). IPF is a 64-bit processor architecture co-developed by Hewlett-Packard Company and Intel Corporation, which is based on Explicitly Parallel Instruction Computing (EPIC). IPF is a well-known family of processors. IPF includes processors such as those having the code names of MERCED, MCKINLEY, and MADISON. In addition to supporting a 64-bit processor bus and a set of 128 registers, the 64-bit design of IPF allows access to a very large memory (VLM) and exploits features in EPIC. While a specific example implementation of one embodiment is described below for the IPF architecture, embodiments of firmware for recovering from LOL described herein are not limited in application to an IPF architecture, but may be applied as well to other architectures (e.g., 32-bit processor architectures, etc.).

Processor architecture generally comprises corresponding supporting firmware, such as firmware 15 of system 10. For example, as described further below in conjunction with the specific example of FIG. 2, the IPF processor architecture comprises such supporting firmware as Processor Abstraction Layer (PAL), System Abstraction Layer (SAL), and Extended Firmware Interface (EFI). Such supporting firmware may enable, for example, the OS to access a particular function implemented for the processor. For instance, the OS may query the PAL as to the size of the cache implemented for the processor, etc. Other well-known functions provided by the supporting firmware (SAL, EFI) include, for example: (a) performing I/O configuration accesses to discover and program the I/O Hardware (SAL_PCI_CONFIG_READ and SAL_PCI_CONFIG_WRITE); (b) retrieving error log data from the platform following a Machine Check Abort (MCA) event (SAL_GET_STATE_INFO); (c) accessing persistent store configuration data stored in non-volatile memory (EFI variable services: GetNextVariableName, GetVariable and SetVariable); and accessing the battery-backed real-time clock/calendar (EFI GetTime and SetTime). Accordingly, the supporting firmware, such as the PAL, is implemented to provide an interface to the processor(s) for accessing the functionality provided by such processor(s). Each of those interfaces provide standard, published procedure calls that are supported. While shown as external to the lockstep processor pair 12 in this example, in certain embodiments all or a portion of firmware 15 may be implemented on a common silicon chip with processors 12A and 12B.

In the example embodiment of FIG. 1, upon firmware 15 being invoked responsive to detection of LOL (by any of error detect logics 13A, 13B, and 13C), firmware 15 determines, in operational block 101, whether the detected LOL is a recoverable LOL. That is, firmware 15 determines in block 101 whether the detected LOL is of a type from which the firmware can recover lockstep for the lockstep processor pair 12 without crashing the system. As described further herein, lockstep is recoverable for certain detected LOLs (which may be referred to as “recoverable LOLs”), while lockstep is not recoverable for other detected LOLs (which may be referred to as “non-recoverable LOLs”). If the lockstep is not recoverable from the detected LOL, then in the example of FIG. 1 firmware 15 crashes the system in block 102.

As described further herein, firmware 15 is implemented in a manner that allows for recovery from certain detected errors without requiring that OS 11 be implemented with specific knowledge for handling such recovery. However, if the lockstep is determined to be recoverable, firmware 15 cooperates

6

with OS 11 via standard OS methods to recover the lockstep. For instance, in the example embodiment of FIG. 1, Advanced Configuration and Power Interface (ACPI) methods are used by firmware 15 to cooperate with OS 11. Accordingly, no processor or platform specific knowledge is required to be embedded in OS 11, but instead any ACPI-compatible OS may be used, including without limitation HP-UX and Open VMS operating systems.

In the example embodiment of FIG. 1, if lockstep is determined to be recoverable in block 101, then firmware 15 triggers OS 11 to idle the master processor 12A in block 103. In this embodiment, firmware 15 utilizes an ACPI method 104 to “eject” master processor 12A, thereby triggering OS 11 to idle the master processor 12A (i.e., stop scheduling tasks for the processor). Of course, by idling master processor 12A, slave processor 12B will in turn be idled. Thus, idling master processor 12A results in idling the lockstep processor pair 12. In this example embodiment, OS 11 is not aware of the presence of slave processor 12B, but is instead aware of master processor 12A. The interface of lockstep processor pair 12 to bus 16 manages copying to slave processor 12B the instructions that are directed by OS 11 to master processor 12A. Thus, firmware 15 need not direct OS 11 to eject slave processor 12B, as OS 11 is not aware of such slave processor 12B in this example implementation. Again, by idling master processor 12A, slave processor 12B is also idled as it merely receives copies of the instructions directed to master processor 12A. Of course, if in a given implementation OS 11 is aware of slave 12B, firmware 15 may be implemented to also direct OS 11 to idle such slave processor 12B in a manner similar to that described for idling master processor 12A. An example technique that may be utilized by firmware 15 for triggering OS 11 to idle master processor 12A in accordance with certain embodiments is described further in concurrently filed and commonly assigned U.S. patent application Ser. No. 10/972,888 titled “SYSTEM AND METHOD FOR SYSTEM FIRMWARE CAUSING AN OPERATING SYSTEM TO IDLE A PROCESSOR”, the disclosure of which is hereby incorporated herein by reference.

Firmware 15 then attempts to recover lockstep for the lockstep processor pair 12 in block 105. For instance, firmware 15 resets the processor pair 12. During such reset of processor pair 12, system 10 can continue to operate on its remaining available processors (not shown in FIG. 1). An example technique that may be utilized by firmware 15 for recovering lockstep in accordance with certain embodiments is described further in concurrently filed and commonly assigned U.S. patent application Ser. No. 10/972,796 titled “SYSTEM AND METHOD FOR MAINTAINING IN A MULTI-PROCESSOR SYSTEM A SPARE PROCESSOR THAT IS IN LOCKSTEP FOR USE IN RECOVERING FROM LOSS OF LOCKSTEP FOR ANOTHER PROCESSOR”, the disclosure of which is hereby incorporated herein by reference.

Once the processor pair 12 is reset and lockstep is recovered, firmware 15 reintroduces master processor 12A to OS 11 in operational block 106. In this embodiment, firmware 15 updates the ACPI device table information for master processor 12A to indicate that such master processor 12A is “present, functioning and enabled.” As discussed in the ACPI 2.0 specification for the _STA status method of a device, the _STA (status) object returns the status of a device, which can be one of the following: enabled, disabled, or removed. In this respect, in the result code (bitmap) bit 0 is set if the device is present; bit 1 is set if the device is enabled and decoding its resources; bit 2 is set if the device should be shown in the UI; bit 3 is set if the device is functioning properly (cleared if the

US 7,502,958 B2

7

device failed its diagnostics); bit 4 is set if the battery is present; and bits 531 are reserved. A device can only decode its hardware resources if both bits 0 and 1 are set. If the device is not present (bit 0 cleared) or not enabled (bit 1 cleared), then the device must not decode its resources. Bits 0, 1 and 3 are the “present, enabled and functioning” bits mentioned above. Firmware 15 utilizes an ACPI method 107 to trigger OS 11 to “check for” master processor 12A, thereby reintroducing the master processor 12A to OS 11. As a result of checking for master processor 12A, OS 11 will recognize that such master processor 12A is again available and will thus begin scheduling tasks for master processor 12A once again. An example technique that may be utilized by firmware 15 for reintroducing a processor to the OS after recovery of lockstep in accordance with certain embodiments is described further in concurrently filed and commonly assigned U.S. patent application Ser. No. 10/973,075 titled “SYSTEM AND METHOD FOR REINTRODUCING A PROCESSOR MODULE TO AN OPERATING SYSTEM AFTER LOCK-STEP RECOVERY”, the disclosure of which is hereby incorporated herein by reference.

FIG. 2 shows a block diagram of one embodiment of the above system 10, which is implemented for the IPF processor architecture and is labeled as system 10A. The quintessential model of the traditional IPF architecture is given in the *Intel IA-64 Architecture Software Developer's Manual, Volume 2: IA-64 System Architecture*, in section 11.1 *Firmware Model*, the disclosure of which is hereby incorporated herein by reference. Accordingly, in this example embodiment of system 10A, firmware 15, labeled as firmware 15A, includes processor abstraction layer (PAL) 201 and platform/system abstraction layer (SAL) 202. In general, PAL 201 is firmware provided by Intel for its processors, and SAL 202 is developed by an original equipment manufacturer (OEM) for the specific system/platform in which the processors are to be employed. PAL 201, SAL 202, as well as an extended firmware interface (EFI) layer (not shown), together provide, among other things, the processor and system initialization for an OS boot in an IPF system.

The boot-up process of a traditional IPF system, for example, proceeds as follows: When the system is first powered on, there are some sanity checks (e.g., power on self-test) that are performed by microprocessors included in the system platform, which are not the main system processors that run applications. After those checks have passed, power and clocks are given to a boot processor (which may, for example, be master processor 12A). The boot processor begins executing code out of the system's Read-Only Memory (ROM) (not specifically shown in FIG. 2). The code that executes is the PAL 201, which gets control of system 10. PAL 201 executes to acquire all of the processors in system 10A (recall that there may be many lockstep processor pairs 12) such that the processors begin executing concurrently through the same firmware.

After it has performed its duty of initializing the processor(s), PAL 201 passes control of system 10A to SAL 202. It is the responsibility of SAL 202 to discover what hardware is present on the system platform, and initialize it to make it available for the OS 11. When main memory is initialized and functional, the firmware 15A is copied into the main memory. Then, control is passed to EFI (not shown), which is responsible for activating boot devices, which typically includes the disk. The EFI reads the disk to load a program into memory, typically referred to as an operating system loader. The EFI loads the OS loader into memory, and then passes it control of system 10A by branching the boot processor into the entry point of such OS loader program.

8

The OS loader program then uses the standard firmware interfaces to discover and initialize system 10A further for control. One of the things that the OS loader typically has to do in a multi-processor system is to retrieve control of the other processors (those processors other than the boot processor). For instance, at this point in a multi-processor system, the other processors may be executing in do-nothing loops. In an ACPI-compatible system, OS 11 makes ACPI calls to parse the ACPI tables to discover the other processors of a multi-processor system in a manner as is well-known in the art. Then OS 11 uses the firmware interfaces to cause those discovered processors to branch into the operating system code. At that point, OS 11 controls all of the processors and the firmware 15A is no longer in control of system 10A.

As OS 11 is initializing, it has to discover from the firmware 15A what hardware is present at boot time. And in the ACPI standards, it also discovers what hardware is present or added or removed at run-time. Further, the supporting firmware (PAL, SAL, and EFI) are also used during system runtime to support the processor. For example, OS 11 may access a particular function of master processor 12A via the supporting firmware 15A, such as querying PAL 201 for the number, size, etc., of the processor's cache 14A. Some other well-known firmware functions that OS 11 may employ during runtime include: (a) PAL 201 may be invoked to configure or change processor features such as disabling transaction queuing (PAL_BUS_SET_FEATURES); (b) PAL 201 may be invoked to flush processor caches (PAL_CACHE_FLUSH); (c) SAL 202 may be invoked to retrieve error logs following a system error (SAL_GET_STATE_INFO, SAL_CLEAR_STATE_INFO); (d) SAL 202 may be invoked as part of hot-plug sequences in which new I/O cards are installed into the hardware (SAL_PCI_CONFIG_READ, SAL_PCI_CONFIG_WRITE); (e) EFI may be invoked to change the boot device path for the next time the system reboots (SetVariable); (f) EFI may be invoked to change the clock/calendar hardware settings; and (g) EFI may be invoked to shutdown the system (ResetSystem).

A “device tree” is provided, which is shown as device tree 203 in this example. Device tree 203 is stored in SRAM (Scratch RAM) on the cell, which is RAM that is reinitialized. Firmware 15A builds the device tree 203 as it discovers what hardware is installed in the system. Firmware then converts this information to the ACPI tables format and presents it to OS 11 so that OS 11 can know what is installed in the system. The ACPI device tables (not shown) are only consumed by OS 11 at boot time, so they are never updated as things change. For OS 11 to find the current status, it calls an ACPI “method” to discover the “current status”. The _STA method described above is an example of such an ACPI method. When _STA is called, the AML can look for properties on the device specified in the firmware device tree and convert that into the Result Code bitmap described above. So, if lockstep has been lost on a processor, firmware 15A will set the device tree property that indicates loss of lockstep, then when OS 11 calls _STA for that device, the “lockstep lost” property directs the AML code to return to “0” in the “functioning properly” bit so that OS 11 can know there is a problem with that processor.

If a lost lockstep is recovered in accordance with the recovery technique described herein, firmware 15A can indicate that lockstep has been recovered in the device tree 203. Then when _STA is called on that device responsive to the OS receiving the “check for device” ACPI method, the present, enabled and functioning bits will all be set and OS 11 will know the CPU is safe to use. A simple example of device tree 203 is shown below in Table 1:

US 7,502,958 B2

9

TABLE 1

Device	Status	Lockstep Enabled
Processor A	Present, Enabled, and Functioning	Yes

Example interactions between the PAL **201**, SAL **202**, and device tree **203** for responding to a detected LOL (e.g., determining it is recoverable and if so then recovering from such LOL) for an IPF system in accordance with certain embodiments are described further herein (e.g., in conjunction with FIG. **5** below), as well as in the co-pending U.S. patent applications incorporated by reference herein.

In certain embodiments, a different recovery technique is employed when LOL is detected for the system's boot processor than is employed for other processors of the system. For various reasons, in certain system architectures problems arise in attempting to idle (or eject) the boot processor from the system. Thus, in certain embodiments, a hot spare processor is used for recovering from LOL for the system's boot processor. More specifically, upon LOL being detected for a boot processor, a hot spare processor (i.e., an idling processor that is available in the system) is transferred the role of boot processor, and then the old boot processor having LOL is reset to reestablish its lockstep. Turning to FIG. **3**, an example system **30** is shown in which firmware utilizes this hot spare technique for recovering from LOL for a boot processor. As with system **10** of FIG. **1**, system **30** includes OS **11**, bus **16**, and lockstep processor pair **12** with its error detect logic **13A-13C**. System **30** further includes a second lockstep processor pair **31** that is communicatively coupled to bus **16**. Lockstep processor pair **31** includes master processor **31A** and slave processor **31B**. Master processor **31A** includes cache **32A**, and slave processor **31B** includes cache **32B**. As with error detect logic **13A-13C** implemented for lockstep processor pair **12**, lockstep processor pair **31** has error detect logic **33A-33C**.

Suppose that during the system boot-up procedure, master processor **12A** assumes the role of boot processor. In this example embodiment, lockstep processor pair **31** may be held as a hot spare for recovering from a LOL that may be detected for the boot processor **12A**. Of course, additional lockstep processor pairs may be included in system **30** (not specifically shown in the example of FIG. **3**), and those additional lockstep processor pairs, which are not the system boot processor, may recover from LOL in the manner described above with FIG. **1**. Thus, hot spare processor pair **31** is not needed for recovering from LOL detected for any non-boot processor, but may instead be used only for recovery of LOL for the boot processor (processor **12A** in this example).

Firmware **35** is included in this example, and upon detection of LOL by any of error detect logics **13A-13C**, it determines whether the lockstep is recoverable in block **101** (as described with FIG. **1** above). If not, firmware **35** crashes the system in block **102** in this example embodiment. If the lockstep is recoverable, then operation advances to block **301** whereat the firmware determines whether the processor for which the LOL was detected is the system's boot processor. Because the LOL is detected for the lockstep processor pair **12** in this example, firmware **35** determines whether master processor **12A** is the system's boot processor. This can be determined, for example, by accessing the device tree **203** of FIG. **2**, as such device tree includes a field for each processor indicating whether such processor is the system's boot processor. This field in the device tree may be set by the firmware

10

during the boot-up process to identify the corresponding processor that is used as the system's boot processor. If determined in block **301** that the master processor **12A** is not the system's boot processor, then operation advances to block **103**, and the lockstep recovery process proceeds in the manner described above with FIG. **1**.

However, if determined in block **301** that the master processor **12A** is the system's boot processor, operation advances to block **302** whereat the state of the "good" processor in the lockstep processor pair **12** is copied over to each of processors **31A** and **31B** in the spare lockstep processor pair **31**. Example techniques that may be utilized by firmware **35** for switching the state of a "good" processor in the lockstep processor pair **12** in accordance with certain embodiments is described further in copending U.S. patent application Ser. No. 10/187,833 (published as publication No. US 2004/0006722 A1) filed Jul. 3, 2002 and titled "METHOD AND APPARATUS FOR RECOVERY FROM LOSS OF LOCK STEP" and in concurrently filed and commonly assigned U.S. patent application Ser. 10/972,588 titled "SYSTEM AND METHOD FOR SWITCHING THE ROLE OF BOOT PROCESSOR TO A SPARE PROCESSOR RESPONSIVE TO DETECTION OF LOSS OF LOCKSTEP IN A BOOT PROCESSOR", the disclosures of which are each hereby incorporated herein by reference.

Thus, operational block **302** essentially makes the spare processor pair **31** the system's boot processor, and then in block **303** firmware **35** resets the lockstep processor pair **12** and reestablishes its lockstep. Once lockstep is reestablished for lockstep processor pair **12**, that pair is held in operational block **304** as a hot spare for the new boot processor pair **31**. That is, firmware **35** updates the device tree **203** of FIG. **2** to reflect that lockstep processor pair **12** is a hot spare for the boot processor pair **31**. Thus, should a recoverable LOL be detected for the new boot processor **31**, the above process may be used by firmware **35** to make the spare lockstep processor pair **12** the boot processor and then recover lockstep for pair **31**. An example technique for resetting the lockstep processor pair **12** and reestablishing its lockstep in which the lockstep processor pair **12** is then held as a hot spare is described further in concurrently filed and commonly assigned U.S. patent application Ser. No. 10/972,796 titled "SYSTEM AND METHOD FOR MAINTAINING IN A MULTI-PROCESSOR SYSTEM A SPARE PROCESSOR THAT IS IN LOCKSTEP FOR USE IN RECOVERING FROM LOSS OF LOCKSTEP FOR ANOTHER PROCESSOR", the disclosure of which is hereby incorporated herein by reference.

One example system in which firmware **35** of FIG. **3** may be employed is shown in FIG. **4**. FIG. **4** shows system **40** that includes multi-processor cells A-D, labeled **41_A-41_D**, respectively. Cell A **41_A** is shown in more detail, and should be understood that cells B-D **41_B-41_D** have substantially the same architecture as that of cell A **41_A** in this example. The architecture of cell A **41_A** described hereafter corresponds to that of Hewlett-Packard's Superdome™ systems. As shown, multi-processor cell A **41_A** includes coherency controller **404** that is communicatively coupled to two buses, labeled **405** and **406**, respectively. A plurality of processors are included within cell A **41_A**, shown as CPU **0**, CPU **1**, CPU **2** and CPU **3** (and labeled **400-403**, respectively). More particularly, a plurality of processors are communicatively coupled to each of the buses **405** and **406**. In this example, CPU **0** (**400**) and CPU **1** (**401**) are each communicatively coupled to the first bus **405**, and CPU **2** (**402**) and CPU **3** (**403**) are each communicatively coupled to the second bus **406**. It should be understood that each of CPU **0** (**400**), CPU **1** (**401**), CPU **2** (**402**),

US 7,502,958 B2

11

and CPU 3 (403) may in actuality be a lockstep processor pair, such as the lockstep processor pair 12 of FIGS. 1-3. Thus, for instance, CPU 0 (400) includes both a master processor and a slave processor, as with processors 12A and 12B included in the lockstep processor pair 12 of FIGS. 1-3.

Coherency controller 404 decodes the address an interrupt is targeted towards and determines which bus (405 or 406) the interrupt packet should be delivered to. Each CPU on the destination bus sees the interrupt and compares the interrupt target address with its internal "LID" register to determine if it is the targeted CPU. If the match is correct, the CPU responds to the interrupt. If the match is not correct, the CPU ignores the interrupt packet.

As shown, the example firmware 35 described above with FIG. 3 may be employed to manage the recovery from detected LOLs for the processors of cells A-D (41_A-41_D). In this example architecture, when designating a hot spare for the boot processor, such hot spare is selected to be another processor communicatively coupled to the same bus as the boot processor. For instance, suppose that during the boot-up of system 40, CPU 2 (402) is designated as the boot processor. Accordingly, firmware 35 will identify another processor that is communicatively coupled to bus 406 to designate as a hot spare for such boot processor, such as CPU 3 (403). During runtime if a LOL is detected for CPU 2 (402), then firmware 35 operates according to the process described above in FIG. 3 to switch the designation of boot processor to the hot spare, CPU 3 (403). In performing this switch the LID register of the CPU that lost lockstep is copied into the LID register of the Hot-Spare CPU and the Hot-Spare CPU pair will then assume the ID of the failed CPU and start responding to interrupts directed to that ID. The LID of the failed CPU is quickly re-written to a different value so that it will no longer respond to interrupts directed to that ID. CPU 2 (402) can then be reset to recover its lockstep, and such CPU 2 (402) may then be held as a hot spare for the new boot processor CPU 3 (403).

It should be recognized that in the example system of FIG. 4, if a hot spare were used for recovering from LOL for all processors, rather than just for the boot processor, much of the potential processing resources would be required to be held as spare resources. For instance, a hot spare would be required on each of the buses of each cell. Because there are two processors coupled to each bus in this example (i.e., CPU 0 and CPU 1 coupled to bus 405, and CPU 2 and CPU 3 coupled to bus 406), reserving a hot spare processor for each bus would result in holding half of the total processing resources in spare. This is an undesirably expensive and wasteful solution. Thus, according to the illustrated embodiment, only a hot spare processor is reserved for the boot processor. Thus, continuing with the above example in which CPU 2 is the boot processor and CPU 3 is held in idle as a hot spare, both of CPU 0 and CPU 1 are active (i.e., neither are held in spare). Further, none of the processors of cells B-D need to be held as hot spares. In the event of a recoverable LOL occurring for any of non-boot processors of the cells, the lockstep recovery process of FIG. 1 (i.e., operational blocks 103-106) can be utilized without requiring a hot spare for lockstep recovery for those non-boot processors.

While in the above example embodiment, a hot spare CPU is maintained for recovery from LOL for the system's boot processor, in an alternative embodiment a hot spare need not be held as such, but rather a processor can be dynamically made a "spare" when needed for recovery from LOL encountered for the system boot processor. That is, assuming as in the above example of FIG. 4 that CPU 2 is the system's boot processor, CPU 3 does not have to be held as a hot spare for CPU 2, but instead CPU 3 can be idled and made a "spare"

12

upon detection of LOL in CPU 2. Thus, the resource of this spare CPU need not be wasted during normal runtime, but can be dynamically turned into a spare processor that is available for recovering from LOL encountered on CPU 2 in the manner described above. As an example of this embodiment, the firmware may send an "eject request" on a healthy processor pair (e.g., CPU 3), and lie to the OS by indicating (e.g., in response to an ACPI_STA method for the healthy processor pair) that such processor pair is not functioning. This would cause the OS to eject that processor pair as though it had encountered a LOL. This processor would become the "hot spare" and be used to replace the boot processor. The system boot processor role would be assumed by the idled healthy processor pair, just as described above in the example of FIG. 4 in which the healthy CPU 3 is held idle for the system boot processor, CPU 2. The only difference being that now the firmware had to take some initial action to turn the healthy processor pair into a "hot spare" by idling it and making it appear to the OS as though it was not functioning. Thus, the LIDs would be swapped. That is, the LID register of the boot processor that lost lockstep is copied into the LID register of the now spare processor pair (CPU 3 in this example), and the LID register of this spare processor pair would be copied into the LID register of the original boot processor. Lockstep is then reestablished on the original boot processor, and it can be reintroduced to the OS with the ID of the original non-boot processor that replaced it as the system boot processor.

Turning now to FIG. 5, an operational flow diagram for one embodiment is shown. In operational block 501, the system is powered on. In operational block 502, the system's firmware establishes a hot spare for the system's boot processor. For instance, in the example system 40 of FIG. 4, if CPU 2 (402) is designated as the boot processor, the firmware 35 identifies a processor on the same bus as the boot processor, such as CPU 3 (403), as a hot spare, and firmware 35 updates the device tree to designate that such CPU 3 (403) is a hot spare for the boot processor CPU 2 (402). An example technique that may be utilized by firmware 35 for establishing a hot spare for the boot processor in accordance with certain embodiments is described further in concurrently filed and commonly assigned U.S. patent application Ser. No. 10/973,077 titled "SYSTEM AND METHOD FOR ESTABLISHING A SPARE PROCESSOR FOR RECOVERING FROM LOSS OF LOCKSTEP IN A BOOT PROCESSOR", the disclosure of which is hereby incorporated herein by reference.

In operational block 503, the firmware establishes lockstep mode on the appropriate processor modules (or lockstep processor pairs). As described further in concurrently filed and commonly assigned U.S. patent application Ser. No. 10/973,004 titled "SYSTEM AND METHOD FOR CONFIGURING LOCKSTEP MODE OF A PROCESSOR MODULE", the disclosure of which is hereby incorporated herein by reference, in certain embodiments whether lockstep is activated for the system (or for any given partition of a system) is configurable by a system administrator. Thus, rather than the processors of a system being fixed in either lockstep mode or not in lockstep mode, certain embodiments permit lockstep mode to be activated and deactivated for a system as may be desired by a system administrator. In the event that lockstep mode is not activated for the system, then the processors that would have been used as slave processors in lockstep mode are available to be used as "master" processors for processing instructions, thus increasing the total amount of computing resources available to the OS. If for example, having a greater number of processors available for computing separate instructions is more desirable than guarding fully against SDC on a given partition of a customer's system, then lock-

US 7,502,958 B2

13

step processing may be disabled for such partition, and lockstep processing may be enabled for those partitions in which fully guarding against SDC is of more importance.

In operational block 504, the boot process boots the system OS and completes with the firmware passing control of the system over to the OS. During system runtime, if the system is operating in lockstep mode, then error detect logic present in the system (such as error detect logic 13A-13C of FIG. 1) monitors for LOL. Thus, block 505 represents the continuous monitoring by the error detect logic of their respective processors for LOL. If an LOL is detected by the error logic, then the firmware is notified of the detected LOL in block 506. An example technique that may be utilized for notifying the firmware of the detected LOL in accordance with certain embodiments is described further in concurrently filed and commonly assigned U.S. patent application Ser. No. 10/972,835 titled "SYSTEM AND METHOD FOR USING INFORMATION RELATING TO A DETECTED LOSS OF LOCKSTEP FOR DETERMINING A RESPONSIVE ACTION", the disclosure of which is hereby incorporated herein by reference. As mentioned above, the firmware determines whether the detected LOL is recoverable. If determined that the LOL is recoverable, then operation continues in the firmware for recovering the LOL in the manner described hereafter.

In block 507, the firmware determines whether the processor for which the LOL is detected is the system's boot processor. This determination may be made by searching for the processor for which the LOL is detected in the device tree (e.g., tree 203 of FIG. 2) and evaluating the fields associated with such processor in the device tree to determine whether the device tree identifies such processor as the boot processor.

If the processor for which LOL is detected is not the system boot processor, then operation advances to block 508 whereat the processor for which the LOL is detected is idled, and actions are taken to reestablish its lockstep. As described above, an ACPI "eject" method may be used to instruct the system's OS to idle the processor, such as described more fully in concurrently filed and commonly assigned U.S. patent application Ser. No. 10/973,003 titled "SYSTEM AND METHOD FOR REESTABLISHING LOCKSTEP FOR A PROCESSOR MODULE FOR WHICH LOSS OF LOCKSTEP IS DETECTED", the disclosure of which is hereby incorporated herein by reference. In order to reestablish lockstep, the processor may be reset. At the end of this step, the processor is executing in firmware space and lockstep has been reestablished for the processor.

The processor is then ready to be introduced back to the OS. Thus, in block 509, the firmware reintroduces to the OS the processor having its lockstep recovered. An example technique that may be utilized for reintroducing the processor back to the OS in block 509 in accordance with certain embodiments is described further in concurrently filed and commonly assigned U.S. patent application Ser. No. 10/973,075 titled "SYSTEM AND METHOD FOR REINTRODUCING A PROCESSOR MODULE TO AN OPERATING SYSTEM AFTER LOCKSTEP RECOVERY", the disclosure of which is hereby incorporated herein by reference. Operation then continues in block 510, wherein the processor whose lockstep was recovered is available to run tasks as scheduled by the OS and the system's processors are monitored for another LOL in block 505. That is, in block 510 the system is back to a stable state with full lockstep protection again being provided.

Unfortunately, as mentioned above, in certain systems the system boot processor that initialized the OS during the systems bootup process cannot easily be idled and returned to

14

firmware control. Thus, a different technique than that described above for nonboot processors is employed in this embodiment. Accordingly, if determined in block 507 that the LOL is detected for the system's boot processor, operation advances to block 511 whereat the state of the "good" boot processor is copied to the hot spare that was established for the boot processor in block 502. As mentioned above, example techniques that may be utilized by the firmware for switching the state of a "good" processor in the lockstep processor pair that is the system boot processor in accordance with certain embodiments is described further in copending U.S. patent application Ser. No. 10/187,833 (published with publication no. US 2004/0006722 A1) filed Jul. 3, 2002 and titled "METHOD AND APPARATUS FOR RECOVERY FROM LOSS OF LOCK STEP" and in concurrently filed and commonly assigned U.S. patent application Ser. No. 10/972,588 titled "SYSTEM AND METHOD FOR SWITCHING THE ROLE OF BOOT PROCESSOR TO A SPARE PROCESSOR RESPONSIVE TO DETECTION OF LOSS OF LOCKSTEP IN A BOOT PROCESSOR", the disclosures of which are each hereby incorporated herein by reference.

In operational block 512, the firmware reestablishes lockstep on the processor for which the LOL was detected (e.g., by resetting the processor), and the firmware establishes the processor having recovered lockstep as the hot spare for the new boot processor. Such processor whose lockstep has been recovered may be established as a spare for the new boot processor in the manner described above in block 502, and as discussed further in concurrently filed and commonly assigned U.S. patent application Ser. No. 10/973,077 titled "SYSTEM AND METHOD FOR ESTABLISHING A SPARE PROCESSOR FOR RECOVERING FROM LOSS OF LOCKSTEP IN A BOOT PROCESSOR", the disclosure of which is hereby incorporated herein by reference. Operation of the system continues in block 513 wherein the processors are monitored for another LOL in block 505. In block 513 the system is back to a stable state with full lockstep protection again being provided.

In certain embodiments, if the lockstep cannot be recovered in operational block 508 (e.g., the processor fails its self-test when being reset), then that processor is simply not reintroduced back to the OS in block 509. A message can be generated for the system administrator to provide notice of the loss of this processor, and if desired the system administrator can schedule an orderly shutdown and service of the system (e.g., for replacing the processor). Similarly, if the lockstep cannot be recovered in operational block 512 (e.g., the processor fails its self-test when being reset), then a message can be generated for the system administrator to provide notice of the loss of this processor (as thus loss of lockstep recovery protection for the boot processor). In other words, if the lockstep cannot be reestablished in block 512, then if a LOL occurs in the new boot processor there will not be a spare processor for taking over its role (and thus the system may crash). If desired, the system administrator can schedule an orderly shutdown and re-boot in order to establish a new boot processor that has a hot spare available and/or schedule service of the system (e.g., for replacing the failed processor).

While a detailed flow diagram of one embodiment is shown in FIG. 5, embodiments hereof are not limited to that example operational flow. For instance, a more general operational flow according to one embodiment is shown in FIG. 6. In FIG. 6, LOL for a pair of processors is detected in operational block 61. In block 62, firmware triggers an OS to idle the processors for which the LOL was detected. In block 63, the firmware recovers lockstep between the pair of processors. In block 64, after lockstep is recovered between the pair of

US 7,502,958 B2

15

processors, the firmware triggers the OS to recognize the processors as again being available for receiving instructions. As described above, the above technique may be modified for recovering from LOL for the boot processor in certain systems.

What is claimed is:

1. A method comprising:
 - detecting loss of lockstep for a lockstep pair of processors; using firmware to trigger an operating system to idle the lockstep pair of processors, recover lockstep for the lockstep pair of processors, and trigger the operating system to recognize the lockstep pair of processors having recovered lockstep; and
 - responsive to said detecting loss of lockstep, using said firmware to determine if lockstep is recoverable for the lockstep pair of processors, wherein said using said firmware to determine if lockstep is recoverable further comprises determining if a lockstep mismatch has occurred between the lockstep pair of processors.
2. The method of claim 1 wherein said using firmware to trigger an operating system to idle the lockstep pair of processors further comprises:
 - using the firmware to trigger the operating system to idle a master processor of the lockstep pair of processors.
3. The method of claim 2 wherein said using firmware to trigger the operating system to recognize the lockstep pair of processors having recovered lockstep further comprises:
 - using the firmware to trigger the operating system to recognize the master of the lockstep pair of processors.
4. The method of claim 1 wherein said detecting loss of lockstep comprises:
 - detecting corrupt data in one of said lockstep pair of processors that would lead to a lockstep mismatch if acted upon.
5. The method of claim 1 wherein said detecting loss of lockstep comprises:
 - detecting a precursor to lockstep mismatch.
6. The method of claim 1 wherein said detecting loss of lockstep comprises:
 - detecting lockstep mismatch between the lockstep pair of processors.
7. The method of claim 1 further comprising:
 - if determined that a lockstep mismatch has not occurred between the lockstep pair of processors, then determining that the loss of lockstep is recoverable.
8. The method of claim 1 further comprising:
 - if determined that a lockstep mismatch has occurred between the lockstep pair of processors, then determining that the loss of lockstep is not recoverable.
9. The method of claim 1 further comprising:
 - determining if said lockstep pair of processors for which said loss of lockstep is detected comprises a system boot processor.
10. The method of claim 9 wherein if determined that said lockstep pair of processors for which said loss of lockstep is detected comprises a system boot processor, further comprising:
 - swapping the role of system boot processor to another processor in the system.
11. The method of claim 10 further comprising:
 - after said swapping, recovering the lockstep for the lockstep pair of processors.
12. The method of claim 11 further comprising:
 - after recovering lockstep for the lockstep pair of processors, holding the recovered pair of processors in idle as a spare for the processor to which the role of system boot processor was swapped.

16

13. A system comprising:
 - an Advanced Configuration and Power Interface (ACPI)-compatible operating system;
 - master processor operable to process instructions scheduled by said ACPI-compatible operating system;
 - slave processor lockstepped with the master processor operable to perform redundant processing of said instructions scheduled for the master processor; and
 - firmware operable, responsive to detection of loss of lockstep between the master and slave processors, to use an ACPI method to trigger the operating system to idle the master processor, attempt to recover lockstep between the master and slave processors, and if recovery of lockstep is successful then use an ACPI method to reintroduce the master processor to the operating system.
14. The system of claim 13 further comprising:
 - error detection logic for detecting loss of lockstep between the master and slave processors.
15. The system of claim 13 further comprising:
 - error detection logic for detecting a precursor to loss of lockstep.
16. The system of claim 15 wherein said precursor to loss of lockstep is treated as said detection of loss of lockstep.
17. The system of claim 15 wherein said error detection logic comprises a parity-based mechanism.
18. The system of claim 13 further comprising error detection logic for detecting lockstep mismatch between the master and slave processors.
19. A system comprising a pair of lockstep processors and computer-executable firmware code stored to a computer-readable medium, the computer-executable firmware code comprising:
 - firmware code, responsive to detection of loss of lockstep between the pair of lockstep processors, for determining if the lockstep is recoverable for the pair of lockstep processors, wherein said firmware code for determining if the lockstep is recoverable further comprises firmware code for determining if a lockstep mismatch has occurred between the pair of lockstep processors; and
 - firmware code, responsive to determining that the lockstep is recoverable, for triggering an operating system to idle the processors, attempting to recover lockstep between the pair of processors, and if lockstep is successfully recovered between the pair of processors, triggering said operating system to recognize the processors as being available for receiving instructions.
20. The system of claim 19 wherein the code for attempting to recover lockstep further comprises:
 - code for resetting the pair of processors.
21. The system of claim 19 wherein said operating system is an Advanced Configuration and Power Interface (ACPI)-compatible operating system, said code for triggering the operating system to idle the processors comprises:
 - code for using an ACPI method to idle the processors.
22. The system of claim 19 wherein said operating system is an Advanced Configuration and Power Interface (ACPI)-compatible operating system, said code for triggering said operating system to recognize the processors as being available for receiving instructions comprises:
 - code for using an ACPI method to cause the operating system to check for the processors.
23. A method comprising:
 - establishing, in a multi-processor system, a hot spare processor for a system boot processor;
 - detecting loss of lockstep for a lockstep pair of processors in said multi-processor system;

US 7,502,958 B2

17

determining if said lockstep pair of processors for which the loss of lockstep is detected is the system boot processor;

if determined that said lockstep pair of processors for which the loss of lockstep is the system boot processor, copying a state of the system boot processor to the hot spare processor; and

if determined that said lockstep pair of processors for which the loss of lockstep is not the system boot processor, then triggering an operating system to a) idle the lockstep pair of processors, b) attempt to recover lockstep between the lockstep pair of processors, and c) if lockstep is successfully recovered between the lockstep pair of processors, trigger said operating system to recognize the processors as being available for receiving instructions.

18

24. The method of claim 23 wherein if determined that said lockstep pair of processors for which the loss of lockstep is the system boot processor further comprising:

attempting to recover lockstep between the lockstep pair of processors after copying the state of the system boot processor to the hot spare processor.

25. The method of claim 24 wherein if determined that said lockstep pair of processors for which the loss of lockstep is the system boot processor further comprising:

if lockstep is successfully recovered between the lockstep pair of processors, establishing the lockstep pair of processors for which lockstep was recovered as a hot spare processor.

* * * * *

UNITED STATES PATENT AND TRADEMARK OFFICE
CERTIFICATE OF CORRECTION

PATENT NO. : 7,502,958 B2
APPLICATION NO. : 10/973076
DATED : March 10, 2009
INVENTOR(S) : Scott L. Michaelis et al.

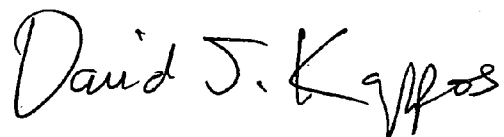
Page 1 of 1

It is certified that error appears in the above-identified patent and that said Letters Patent is hereby corrected as shown below:

In column 7, line 2, delete “531” and insert -- 5-31 --, therefor.

Signed and Sealed this

Eighteenth Day of August, 2009

A handwritten signature in black ink that reads "David J. Kappos". The signature is written in a cursive, flowing style.

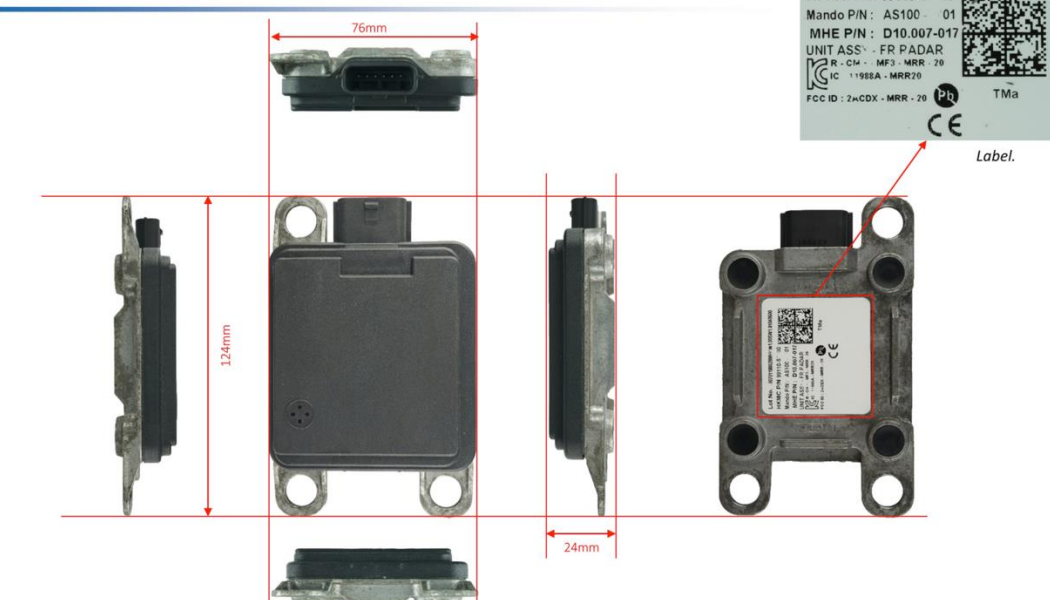
David J. Kappos
Director of the United States Patent and Trademark Office

U.S. Patent No. 7,502,958 (“’958 Patent”)**Accused Products**

HL Klemove products, including without limitation its Advanced Driver Assistance Systems (ADAS), which include a Mando MRR-20 Mid-Range Radar (containing an Infineon SAK-TC297TA chipset) and/or Hella RS4 Lange Change Assist Radar (containing an Infineon SAK-TC264DA chipset) (“Accused Products”), infringe at least claim 19 of the ’958 Patent.

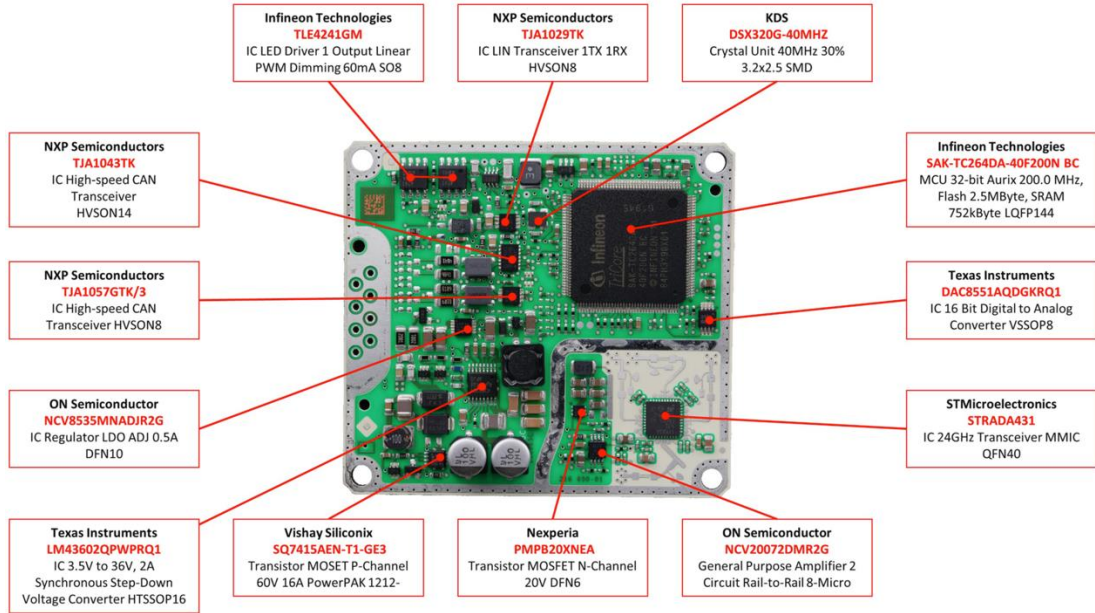
Claim 19

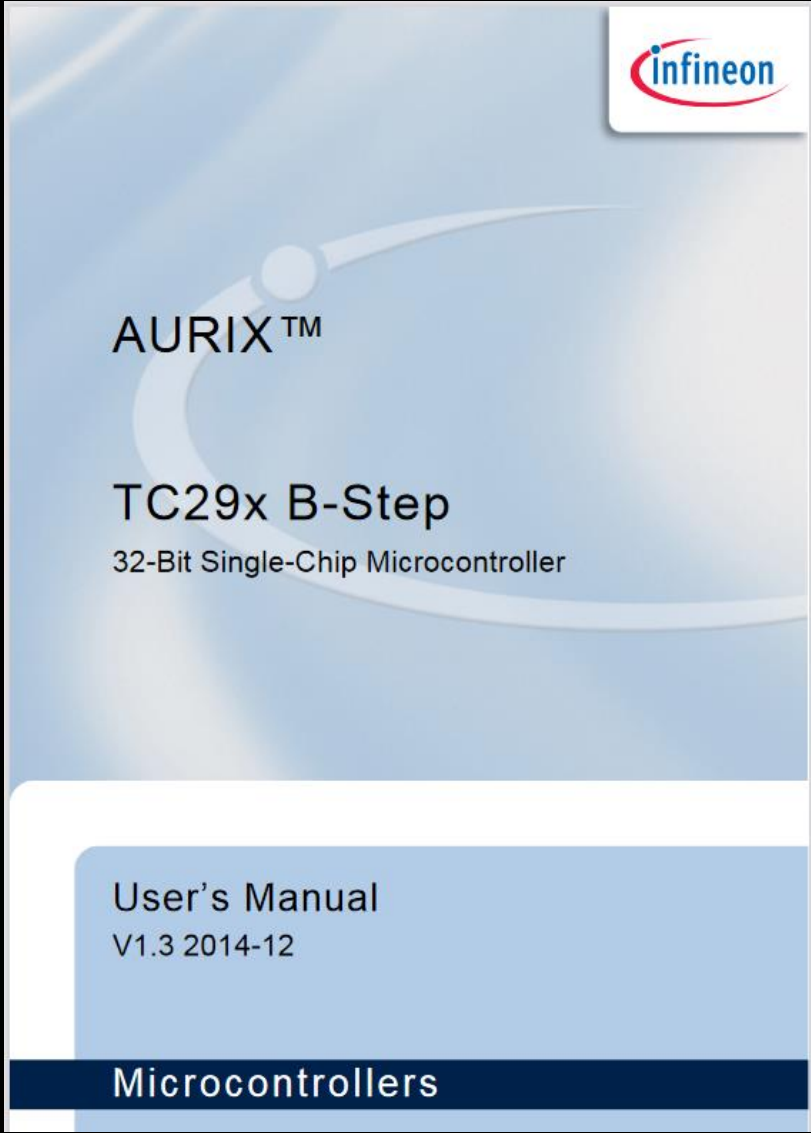
Claim 19	Exemplary Evidence of Infringement
[19-Pre] A system comprising a pair of lockstep processors and computer-executable firmware code stored to a computer-readable medium, the computer-executable firmware code comprising:	The Accused Products contain a Mid-Range Radar and/or Lange Change Assist Radar comprising a pair of lockstep processors and computer-executable firmware code stored to a computer-readable medium. For example, the Accused Products use the Infineon TriCore SAK-TC297TA system-on-a-chip (SoC) in their Mid-Range Radar:

Claim 19	Exemplary Evidence of Infringement
	Views and Dimensions  Label. Lot No. : 00701180529M21W1.005W1.01049600 HKMC P/N : 99110-S* 00 Mando P/N : AS100 - 01 MHE P/N : D10.007-017 UNIT ASS* - FR PADAR R - CM - MF3 - MRR - 20 IC 1198EA - MRR20 FCC ID : 2ACDX - MRR - 20 CE TMa

Claim 19	Exemplary Evidence of Infringement
	Electronic Board – Top Side – Main Components Identification For example, the Accused Products use the Infineon SAK-TC264DA system-on-a-chip (SoC) in their Lange Change Assist Radar:

Claim 19	Exemplary Evidence of Infringement
	

Claim 19	Exemplary Evidence of Infringement
	Electronic Board – Top Side – Main Components Identification  On information and belief, the operation and functionality of the Infineon TC29TA SoC is representative of the operation and functionality of the Infineon TC264DA SoC.

Claim 19	Exemplary Evidence of Infringement
	 https://www.infineon.com/dgdl/Infineon-TC29x_B-step-UM-v01_03-

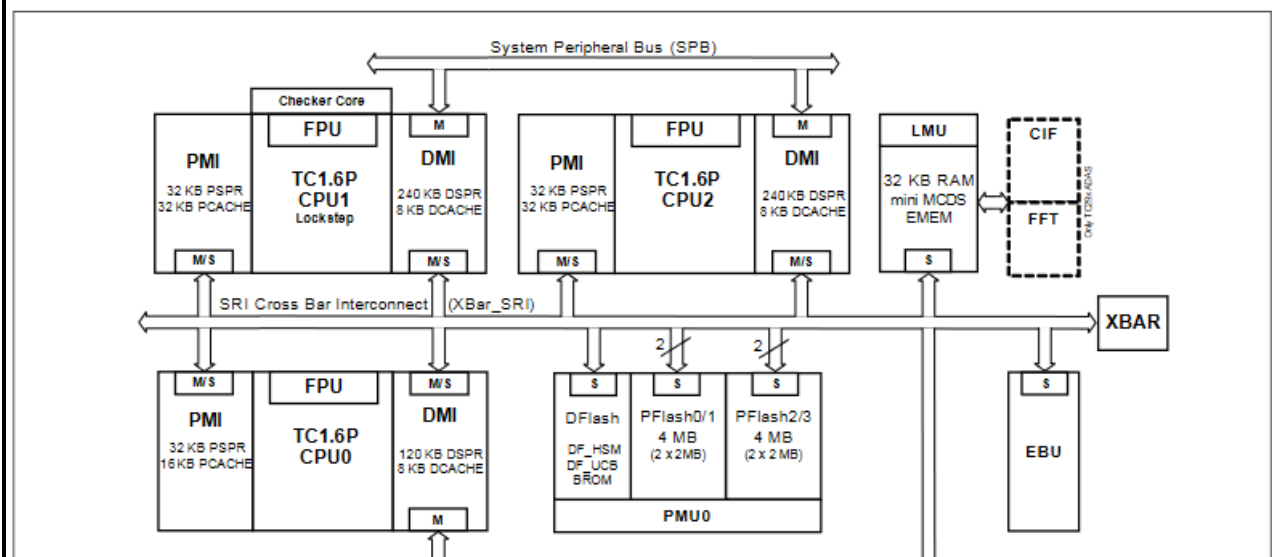
Claim 19**Exemplary Evidence of Infringement**

[EN.pdf?fileId=5546d46269bda8df0169ca1bdee424a2](#); see also https://www.infineon.com/dgdl/Infineon-TC26x_B-step-UM-v01_03-EN.pdf?fileId=5546d46269bda8df0169ca09970023e8

The TC29x processors at least one pair of lockstep processors. For example, the block diagram of the TC29x shown below includes a pair of lockstep processors (CPU0 and CPU1) and non-volatile program memory.

1.2.1 TC29x Block Diagram

Figure 1 shows the block diagram of the TC29x. Please note that not all features that are shown in the block diagram are available in all TC29x package variants.



Id., pg. 1-9

5.12.8 Lock Step Implementation

The TC1.6E and TC1.6P CPUs may be implemented as standard CPU (as described in the chapter) or as a checker CPU for use in lockstep comparison system. The checker CPU implementation has the following features.

- No Memories

User's Manual
CPU V3.7

5-113

V1.3, 2014-12



TC29x B-Step

CPU Subsystem

- All signals that would normally be fed to the memories become additional outputs from the checker CPU.
- All signals that would normally be fed from the memories become additional inputs to the checker CPU.
- All outputs from the checker CPU are inverted.

The lockstep system compares the following CPU interfaces.

- Interface to the interrupt router.
- Interface to the SCU.
- Interface to the DMI Data Scratch pad memory.
- Interface to the PMI Program Scratch memory.
- Interface to the PMI Program Cache memory.
- Interface to the PMI Program Tag memory.
- Master interface to the Peripheral bus.
- Master interface to the SRI bus for PMI accesses.
- Master interface to the SRI bus for DMI accesses.
- Slave accesses from the SRI bus for PMI and DMI accesses.
- The CPUs run in lockstep mode at the full specified frequency. Enabling lockstep does not require operational frequency to be reduced.
- The Checker CPU is implemented using inverse state storage "AntiCore".

Claim 19	Exemplary Evidence of Infringement
	<i>Id.</i> pg. 5-113 – 5-114 6 Lockstep Comparator Logic (LCL) The Lockstep Comparator Logic module provides access to the control and self test functions of the processor lockstep comparators. 6.1 Feature List An overview of the features implemented in the LCL follows: • Monitoring the core comparators for the lockstep core and its shadow and flagging any detected differences • Running background, continuous self test on the lockstep comparators to validate the correct operation of the logic. <i>Id.</i> pg. 6-1

Claim 19	Exemplary Evidence of Infringement
	10 Program Memory Unit (PMU) The Program Memory Unit (PMU) controls the Flash memory and the BootROM and connects them to the SRI crossbar. The devices of the AURIX family have at least one PMU. This is named "PMU0". Depending on the amount of Flash memory more PMUs are added which are named "PMU1", and so on. All PMUs are independent from each other. TC29x has 1 PMU(s). Throughout this document a generic PMU is specified. The Chapter 10.2 lists the configuration parameters of each PMU and their values in the TC29x. When PMU functionality depends on a parameter this is indicated in the text. This chapter has the following structure: • Generic feature list and block diagram (Chapter 10.1). • PMU configuration of the TC29x (Chapter 10.2). • Functionality of the BootROM (Chapter 10.3). • Functionality of the tuning protection (Chapter 10.4). • Functionality of the Flash memory (Chapter 10.5). • Signaling to the Safety Management Unit "SMU" (Chapter 10.6). • Register Set (Chapter 10.7). • Application Hints (Chapter 10.8). <i>Id.</i> pg. 10-1

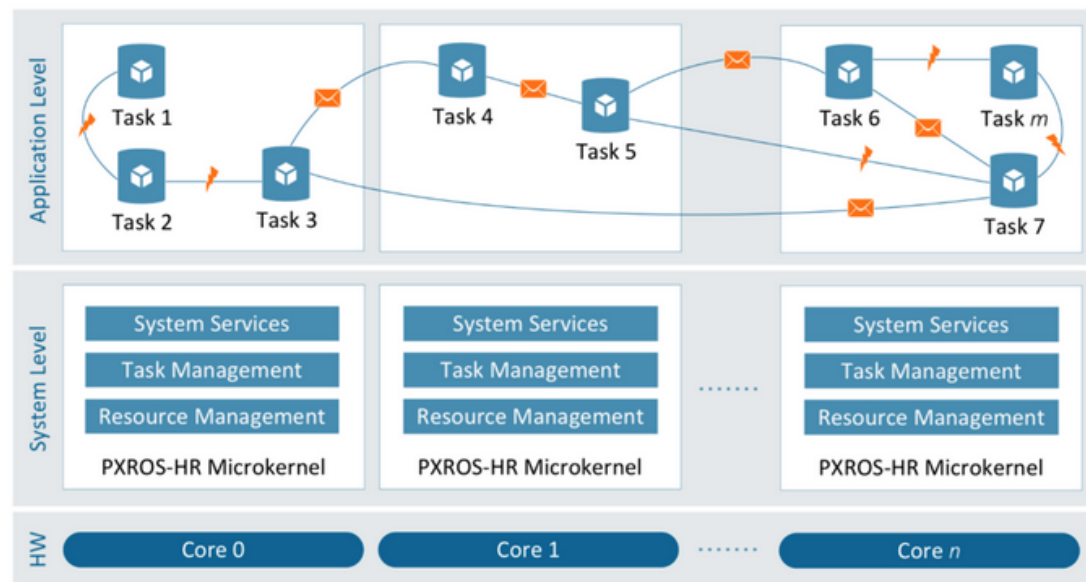
Claim 19	Exemplary Evidence of Infringement
	Program Memory Unit (PMU) Figure 10-1 PMU/FSI/Flash Basic Block Diagram
[19a] firmware code, responsive to detection of loss of lockstep between the pair of lockstep	<i>Id.</i> pg. 10-2 The Accused Products comprise firmware code, responsive to detection of loss of lockstep between the pair of lockstep processors, for determining if the lockstep is recoverable for the pair of lockstep processors, wherein said firmware code for determining if the lockstep is recoverable further comprises firmware code for determining if a lockstep mismatch has occurred between the pair of lockstep processors.

Claim 19	Exemplary Evidence of Infringement
processors, for determining if the lockstep is recoverable for the pair of lockstep processors, wherein said firmware code for determining if the lockstep is recoverable further comprises firmware code for determining if a lockstep mismatch has occurred between the pair of lockstep processors; and	For example, the Infineon TriCore TC29X based systems necessarily include firmware code. On information and belief, the Accused Products use an ASIL D (ISO26262) operating system such as the Hitex PXROS Real-time OS for TriCore and Aurix processors such as the TC29X.

Claim 19**Exemplary Evidence of Infringement**

PXROS - Real-time OS for TriCore™ and AURIX™

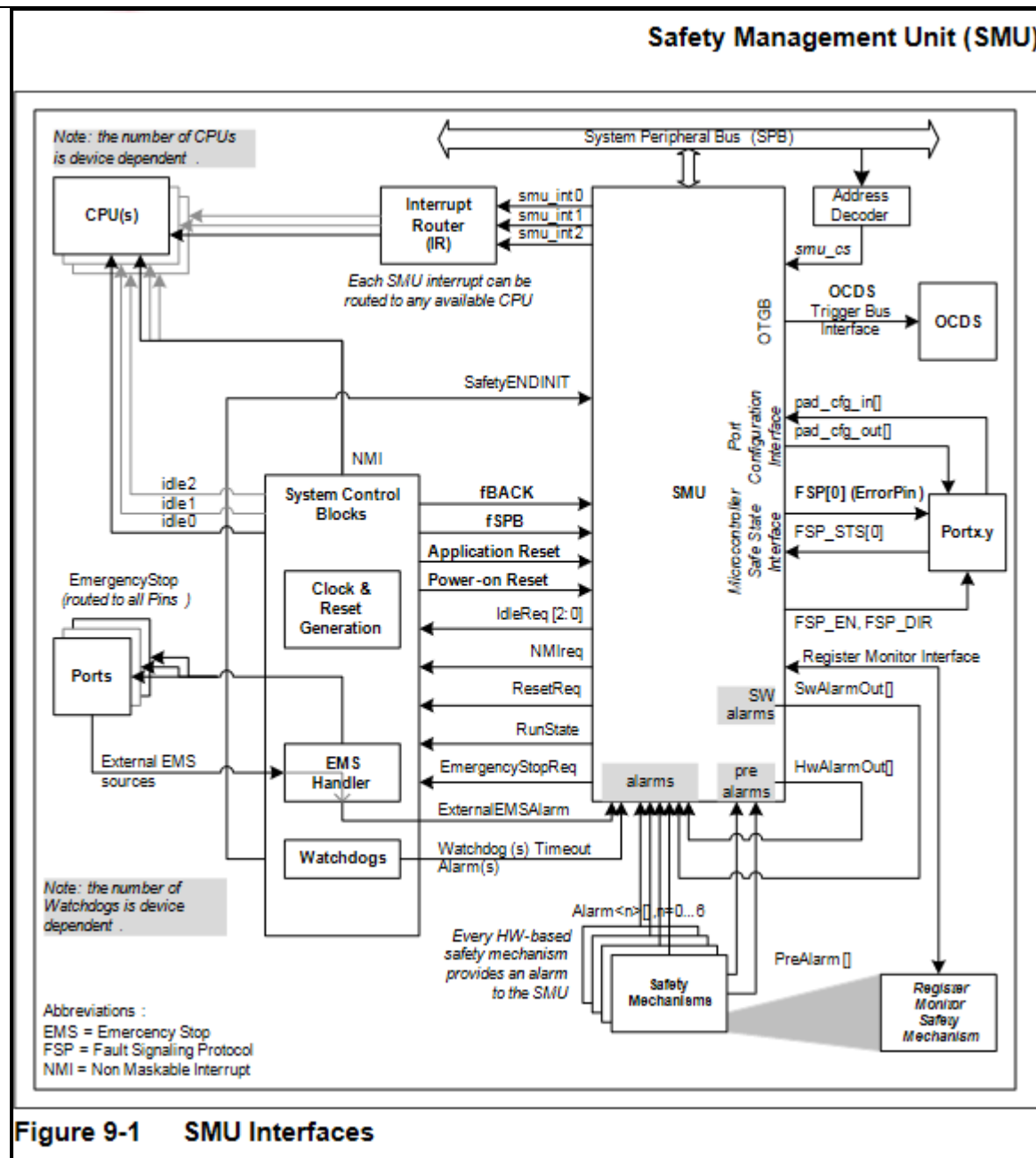
PXROS-HR is an object-oriented real-time operating system (RTOS) with a very modern microkernel. Its architecture makes it particularly suitable for use on modern multicore MCUs. The current version of PXROS-HR brings improvements in the concepts for encapsulation and has improved robustness. This is achieved by using hardware protection mechanisms that are part of modern microcontrollers such as Infineon's AURIX family.



<https://www.hitex.com/tools-components/middleware/rtos/pxros-real-time-os-for-tricore-and-aurix>

	Safety approved and certified The PXROS-HR operating system for TriCore is officially safety tested. The certificate on the suitability of PXROS-HR for safety-relevant applications up to SIL 3 (IEC61508) and ASIL D (ISO 26262) is valid from February 2019 to February 2024. Part of the certification is an evaluation report with the results of the certification by TÜV-Nord Systems GmbH & Co. KG. PXROS-HR was developed on the basis of the HighTec C/C++ compiler for TriCore/AURIX. This makes it particularly suitable for safety-relevant industrial and automotive applications. It integrates into the software frameworks MCAL and SafeTlib from Infineon, is highly optimized for the TriCore architecture and offers multi-core support for the AURIX family (... read also about the Hitex SafeTpack for AURIX). Main features • Fast real-time microkernel operating system • Message-based intermediate tasks and intermediate cores • MPU-protected task resources and message data • Enhanced resource and rights management for tasks • Low latency flexible interrupt subsystem • Meets the requirements of IEC 61508 (SIL3) and ISO 26262 (ASIL D) Essential design elements • Freedom from interference in the granularity of the task-level • Core-agnostic task code and communication scheme • Identical application patterns for single and multicore designs • Simple single-to-multicore migration • Reusability of software components (task code) • Simplified core load distribution and design verification	
--	--	--

Claim 19	Exemplary Evidence of Infringement
	<i>Id.</i> For example, the Lockstep Comparator Logic (LCL) compares the outputs from the master and checker cores and reports a failure to the Safety Management Unit (SMU): 6.3 Lockstep Monitoring The lockstep monitoring function will compare the outputs from the master and checker cores and report that a failure has occurred to the Safety Management Unit (SMU) for appropriate action. The monitoring function temporally separates the cores by inserting synchronisation delays to re-align the signals being compared. To achieve this, the checker core inputs and the master core outputs fed to the comparators are delayed by two clock cycles. https://www.infineon.com/dgdl/Infineon-TC29x_B-step-UM-v01_03-EN.pdf?fileId=5546d46269bda8df0169ca1bdee424a2 pg. 6-1  TC29x B-Step Safety Management Unit (SMU) 9.2 Features The SMU implements the following features: • Collects every alarm (error notification) information from every safety mechanism: – CPU Lockstep <i>Id.</i> pg. 9-3 For example, the SMU can be configured to generate a signal to the firmware indicating loss of lockstep.



Claim 19	Exemplary Evidence of Infringement									
	Id. pg. 9-2 - Each individual alarm can be configured to activate the fault signaling protocol and/or one of the following internal actions: User's Manual SMU, 3.7 9-3 V1.3, 2014-12  TC29x B-Step Safety Management Unit (SMU) – generate an Interrupt request to any of the CPUs, concurrent interrupts to several CPUs can be configured – generate a NMI request to the System Control Unit – generate a reset request to the System Control Unit leading to a system or application reset – activate the pad emergency stop signal controlling the safe state of output pads – generate a CPU idle request Id. pg. 9-3 – 9-4 <table><tr><th colspan="3">Table 9-17 TC29x Alarm Mapping related to Alarm 0 group</th></tr><tr><th>Alarm Index</th><th>Module</th><th>Safety Mechanism & Alarm Indication</th></tr><tr><td>ALM0[0]</td><td>CPU0</td><td>Safety Mechanism: Lockstep CPU Alarm: Lockstep Comparator Error</td></tr></table> Id. pg. 9-25	Table 9-17 TC29x Alarm Mapping related to Alarm 0 group			Alarm Index	Module	Safety Mechanism & Alarm Indication	ALM0[0]	CPU0	Safety Mechanism: Lockstep CPU Alarm: Lockstep Comparator Error
Table 9-17 TC29x Alarm Mapping related to Alarm 0 group										
Alarm Index	Module	Safety Mechanism & Alarm Indication								
ALM0[0]	CPU0	Safety Mechanism: Lockstep CPU Alarm: Lockstep Comparator Error								

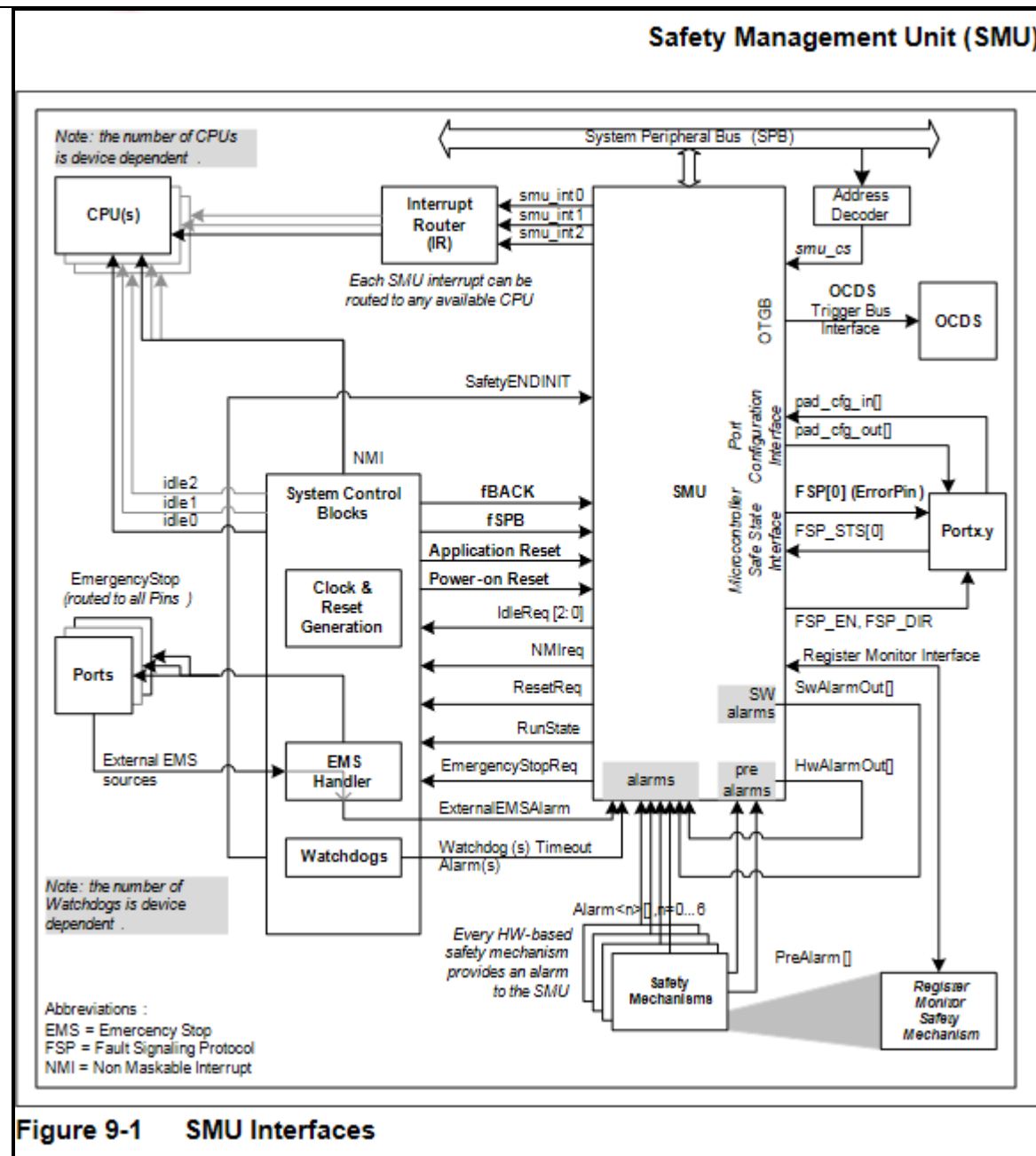
Claim 19	Exemplary Evidence of Infringement						
	Table 9-18 TC29x Alarm Mapping for related to ALM1 group <table><tr><th>Alarm Index</th><th>Module</th><th>Description</th></tr><tr><td>ALM1[0]</td><td>CPU1</td><td>Safety Mechanism: Lockstep CPU Alarm: Lockstep Comparator Error</td></tr></table> <i>Id.</i> pg. 9-28 For example, the firmware may detect if the lockstep loss is recoverable by, e.g. clearing the lockstep alarm and determining if the lockstep alarm gets set again or remains clear. 9.4.8.4 FSP Fault State When an alarm configured to activate the FSP, the SMU automatically switches to the FAULT state. During this time it is also possible for the safety-related software to try to analyze the root cause (when the microcontroller is still operational) and decide about the severity of the error. As the FSP is active for at least T_{FSP_FS} (see SMU_FSP register), it is ensured that the safe state of the system is entered through external mechanisms independent from the microcontroller (besides FSP itself). <i>Id.</i> pg. 9-56	Alarm Index	Module	Description	ALM1[0]	CPU1	Safety Mechanism: Lockstep CPU Alarm: Lockstep Comparator Error
Alarm Index	Module	Description					
ALM1[0]	CPU1	Safety Mechanism: Lockstep CPU Alarm: Lockstep Comparator Error					

SMU_STS																Reset Value: 0000 0000 _H			
Status Register (24 _H)																			
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16				
0												RTM E1	RTS 1	RTM E0	RTS 0				
r												rwh	rwh	rwh	rwh				
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0				
0		FST S		FSP		ASC E		RES		ARG		CMD							
r		rwh		rh		rwh		rwh		rwh		rwh							
Field	Bits	Type	Description																
CMD	[3:0]	rwh	Last command received Same encoding as CMD field of SMU_CMD register																
ARG	[7:4]	rwh	Last command argument received Same encoding as ARG field of SMU_CMD register																
RES	8	rwh	Result of last received command 0 _B Command was successful 1 _B Command failed																
ASCE	9	rwh	Alarm Status Clear Enable This bit controls if a status flag set in an AG<x> register upon detection of an alarm event can be cleared by software or not. When ASCE is enabled software shall write a 1 to the bit position in AG<x> to clear the bit (W1C). When a W1C action takes place the ASCE bit is automatically cleared to 0 by hardware and software shall set the ASCE bit again by using the SMU_ASCE() command. 0 _B Alarm status bits AG<x> can not be cleared 1 _B Alarm status bits AG<x> can be cleared.																

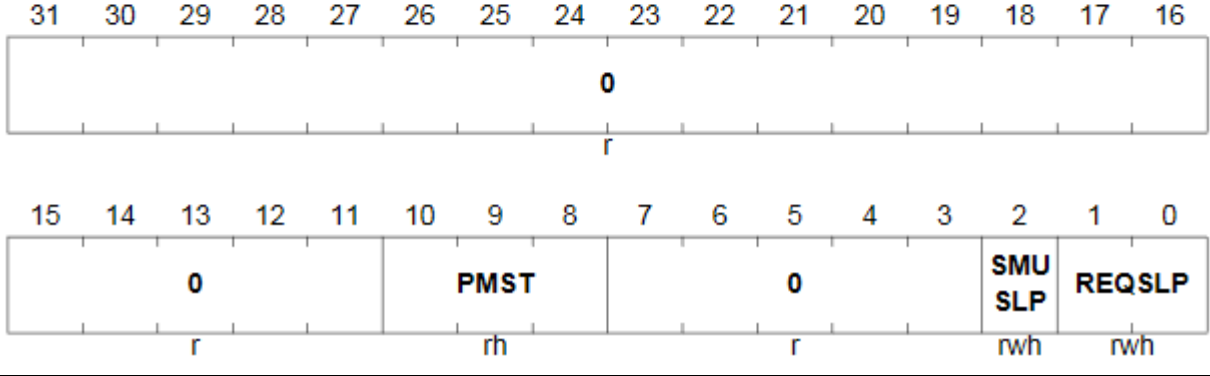
Claim 19	Exemplary Evidence of Infringement																																																																																																								
	<i>Id.</i> pg. 9-80 9.5.5 SMU Alarm Status Registers SMU Alarm status registers related to the different alarm groups. SMU_AG_x (x=0-6) Alarm Status Register (1C0_H + x*04_H) Reset Value: 0000 0000_H <table><tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr><tr><td>SF31</td><td>SF30</td><td>SF29</td><td>SF28</td><td>SF27</td><td>SF26</td><td>SF25</td><td>SF24</td><td>SF23</td><td>SF22</td><td>SF21</td><td>SF20</td><td>SF19</td><td>SF18</td><td>SF17</td><td>SF16</td></tr><tr><td>rwh</td><td>rwh</td><td>rwh</td><td>rwh</td><td>rwh</td><td>rwh</td><td>rwh</td><td>rwh</td><td>rwh</td><td>rwh</td><td>rwh</td><td>rwh</td><td>rwh</td><td>rwh</td><td>rwh</td><td>rwh</td></tr><tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr><tr><td>SF15</td><td>SF14</td><td>SF13</td><td>SF12</td><td>SF11</td><td>SF10</td><td>SF9</td><td>SF8</td><td>SF7</td><td>SF6</td><td>SF5</td><td>SF4</td><td>SF3</td><td>SF2</td><td>SF1</td><td>SF0</td></tr><tr><td>rwh</td><td>rwh</td><td>rwh</td><td>rwh</td><td>rwh</td><td>rwh</td><td>rwh</td><td>rwh</td><td>rwh</td><td>rwh</td><td>rwh</td><td>rwh</td><td>rwh</td><td>rwh</td><td>rwh</td><td>rwh</td></tr></table><table><thead><tr><th>Field</th><th>Bits</th><th>Type</th><th>Description</th></tr></thead><tbody><tr><td>SFn (n = 0-31)</td><td>n</td><td>rwh</td><td>Status flag for alarm n belonging to alarm group x (x=0-6). 0_B Status flag n does not report a fault condition 1_B Status flag n reports a fault condition</td></tr></tbody></table> <i>Id.</i> pg. 9-115 As another example of how the firmware code may determine if the lockstep is recoverable, the firmware may test the functionality of the lockstep comparator:	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	SF31	SF30	SF29	SF28	SF27	SF26	SF25	SF24	SF23	SF22	SF21	SF20	SF19	SF18	SF17	SF16	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	SF15	SF14	SF13	SF12	SF11	SF10	SF9	SF8	SF7	SF6	SF5	SF4	SF3	SF2	SF1	SF0	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	Field	Bits	Type	Description	SFn (n = 0-31)	n	rwh	Status flag for alarm n belonging to alarm group x (x=0-6). 0 _B Status flag n does not report a fault condition 1 _B Status flag n reports a fault condition
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16																																																																																										
SF31	SF30	SF29	SF28	SF27	SF26	SF25	SF24	SF23	SF22	SF21	SF20	SF19	SF18	SF17	SF16																																																																																										
rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh																																																																																										
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0																																																																																										
SF15	SF14	SF13	SF12	SF11	SF10	SF9	SF8	SF7	SF6	SF5	SF4	SF3	SF2	SF1	SF0																																																																																										
rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh																																																																																										
Field	Bits	Type	Description																																																																																																						
SFn (n = 0-31)	n	rwh	Status flag for alarm n belonging to alarm group x (x=0-6). 0 _B Status flag n does not report a fault condition 1 _B Status flag n reports a fault condition																																																																																																						

Claim 19	Exemplary Evidence of Infringement
	6.5 Lockstep Failure Signalling Test The lockstep comparator allows a failure to be injected into one of the comparator nodes to allow the signalling of failures to be verified. A failure can be injected by writing 1_B to the LCLT1 bitfield of the LCLTEST register. The failure will be injected for a single cycle of the SPB clock. It is not necessary to write a 0_B to clear the test. <i>Id.</i> pg. 6-4 The TC29X includes support for recovery from safety alarm warnings such as loss of lockstep: A watchdog may be used within a safety application to provide a recovery time period during which software might attempt to recover from a safety alarm warning. To ensure that a CPU fault could not allow a fault to be ignored an option is provided to prevent watchdog unlocking if the Safety Management Unit (SMU) is not in the RUN state. This option may be enabled by bit WDTxCON0.UR. <i>Id.</i> pg. 7-270 Implements an internal watchdog called recovery timer to monitor the execution of critical software error handlers. The watchdog is started automatically by hardware according to configurable alarm events. <i>Id.</i> pg. 9-4
[19b] firmware code, responsive to determining that the lockstep is recoverable, for triggering an operating system to idle the processors, attempting to recover lockstep between the pair of processors, and if lockstep is successfully	The Accused Products comprise firmware code, responsive to determining that the lockstep is recoverable, for triggering an operating system to idle the processors, attempting to recover lockstep between the pair of processors, and if lockstep is successfully recovered between the pair of processors, triggering said operating system to recognize the processors as being available for receiving instructions. For example, the firmware code, in response to determining that the lockstep is recoverable, may trigger an operating system to idle the processors. For example, the lockstep processors may be put into an idle state by the SMU upon detection of a lockstep mismatch.

Claim 19	Exemplary Evidence of Infringement
recovered between the pair of processors, triggering said operating system to recognize the processors as being available for receiving instructions.	 TC29x B-Step Safety Management Unit (SMU) – generate an Interrupt request to any of the CPUs, concurrent interrupts to several CPUs can be configured – generate a NMI request to the System Control Unit – generate a reset request to the System Control Unit leading to a system or application reset – activate the pad emergency stop signal controlling the safe state of output pads – generate a CPU idle request <i>Id.</i> pg. 9-4



Claim 19	Exemplary Evidence of Infringement						
	<i>Id.</i> pg. 9-2 As a further example, the lockstep processors may be idled or reset by firmware running on a CPU, by e.g. writing to the SCU's PMCSR register for the appropriate CPU. 7.3.2.1 Power Management Overview The Power Management scheme allows activation of power down modes so that the system operates with the minimum required power for the corresponding application state. A progressive reduction in power consumption is achieved by invoking Idle, Sleep or Standby modes respectively. The Idle mode is specific to each individual CPU where as Sleep and Standby modes influence the complete system. As shown in Table 7-20, there are two power modes available for each CPU: • CPU Run Mode • CPU Idle Mode Table 7-20 CPU Power Management <table data-bbox="583 889 1780 1401"> <tr> <th data-bbox="583 889 772 930">Mode</th><th data-bbox="772 889 1780 930">Description</th></tr> <tr> <td data-bbox="583 938 772 979">Run Mode</td><td data-bbox="772 938 1780 979">The CPU clock is active and code is being executed.</td></tr> <tr> <td data-bbox="583 987 772 1027">Idle Mode</td><td data-bbox="772 987 1780 1401"> CPU may enter Idle Mode on following events: • On a SW Idle request issued by setting register bits PMCSR_x.REQSLP = 01_B when CPU has no active tasks to perform. • On a SW Idle request (PMCSR_y.REQSLP = 01_B) issued by another CPU. • On a SMU Idle request in case CPU faults are detected triggering safety alarms which are configured in SMU to set CPU into Idle state. The CPU code execution is halted and CPU clock is disabled in Idle state. The peripherals continue to remain active. CPU RAM memories (PSPR / DSPR) are accessible to other bus masters and peripherals. CPU may exit Idle mode on following events: </td></tr> </table>	Mode	Description	Run Mode	The CPU clock is active and code is being executed.	Idle Mode	CPU may enter Idle Mode on following events: • On a SW Idle request issued by setting register bits PMCSR_x.REQSLP = 01_B when CPU has no active tasks to perform. • On a SW Idle request (PMCSR_y.REQSLP = 01_B) issued by another CPU. • On a SMU Idle request in case CPU faults are detected triggering safety alarms which are configured in SMU to set CPU into Idle state. The CPU code execution is halted and CPU clock is disabled in Idle state. The peripherals continue to remain active. CPU RAM memories (PSPR / DSPR) are accessible to other bus masters and peripherals. CPU may exit Idle mode on following events:
Mode	Description						
Run Mode	The CPU clock is active and code is being executed.						
Idle Mode	CPU may enter Idle Mode on following events: • On a SW Idle request issued by setting register bits PMCSR_x.REQSLP = 01_B when CPU has no active tasks to perform. • On a SW Idle request (PMCSR_y.REQSLP = 01_B) issued by another CPU. • On a SMU Idle request in case CPU faults are detected triggering safety alarms which are configured in SMU to set CPU into Idle state. The CPU code execution is halted and CPU clock is disabled in Idle state. The peripherals continue to remain active. CPU RAM memories (PSPR / DSPR) are accessible to other bus masters and peripherals. CPU may exit Idle mode on following events:						

Claim 19	Exemplary Evidence of Infringement
	<i>Id.</i> pg. 7-195 7.3.2.5 Power Management Registers Power Management Control and Status Register The set of registers used for Power Management control the issue of power modes, manage wake-up configuration and provide status information on mode transitions. The request for Idle, Sleep or Standby mode is issued via PMCSR_x registers. PMCSR0 Power Management Control and Status Register (0D4_H) Reset Value: 0000 0100_H PMCSR1 Power Management Control and Status Register (0D8_H) Reset Value: 0000 0100_H PMCSR2 Power Management Control and Status Register (0DC_H) Reset Value: 0000 0100_H  <i>Id.</i> pg. 7-210

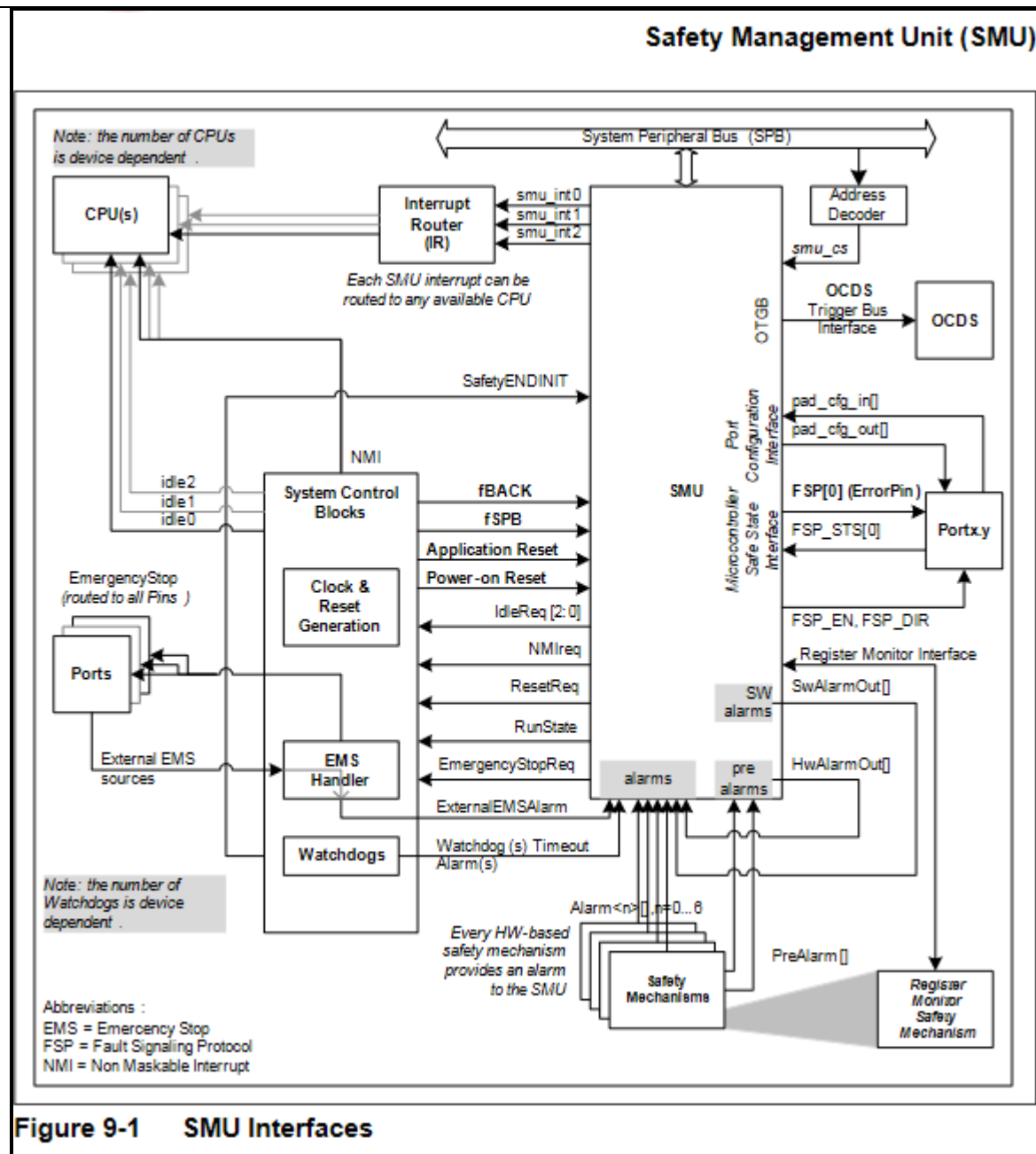
Claim 19	Exemplary Evidence of Infringement			
	Field REQSLP	Bits [1:0]	Type rwh	Function Idle Mode and Sleep Mode Request 00 _B Request CPU Run Mode 01 _B Request CPU Idle Mode 10 _B Request System Sleep Mode 11 _B Request System Standby Mode In Idle Mode or Sleep Mode, these bits are cleared in response to an interrupt for the CPU, or when bit 15 of the corresponding CPU Watchdog Timer register (bit WDTCPUxSR.TIM[15]) changes from 0 to 1. In Standby Mode, these bits are cleared on wake-up. REQSLP maybe written only when either CPU or Safety ENDINIT bits are set to 0. CPU ENDINIT bit has to be set back after REQSLP is written for the mode transition to take place. In case of Safety ENDINIT, the mode transition will be issued immediately and does not wait till Safety ENDINIT is set back to 1 again.
	<i>Id.</i> pg. 7-211 For example, the firmware may attempt to recover lockstep operation by, e.g. resetting the processors:			

Claim 19	Exemplary Evidence of Infringement
	SW Reset Configuration Register This register controls the SW Reset operation. As for other kernel registers, write access to the SWRSTCON register is configurable via SCU_ACCEN0 and additionally is temporally restricted by Safe ENDINIT. These restrictions can be used to provide protection against accidental reset by unauthorised CPUs or system bus masters. SWRSTCON Software Reset Configuration Register (060_H)Reset Value: 0000 XX00_H 31302928272625242322212019181716 0 r 1514131211109876543210 RES 0 SW RST REQ 0 rw r w r
	Id. pg. 7-112

Claim 19	Exemplary Evidence of Infringement																							
	<table><tr><th>Field</th><th>Bits</th><th>Type</th><th>Description</th></tr><tr><td>0</td><td>0</td><td>rw</td><td>Reserved Read as 0; should be written with 0.</td></tr><tr><td>SWRSTREQ</td><td>1</td><td>w</td><td>Software Reset Request 0_B No SW Reset is requested 1_B A SW Reset request trigger is generated This bit is automatically cleared and read always as zero.</td></tr><tr><td>RES</td><td>[15:8]</td><td>rw</td><td>Reserved Should be written with original content.</td></tr><tr><td>0</td><td>[7:2], [31:16]</td><td>r</td><td>Reserved Read as 0; should be written with 0.</td></tr></table>				Field	Bits	Type	Description	0	0	rw	Reserved Read as 0; should be written with 0.	SWRSTREQ	1	w	Software Reset Request 0 _B No SW Reset is requested 1 _B A SW Reset request trigger is generated This bit is automatically cleared and read always as zero.	RES	[15:8]	rw	Reserved Should be written with original content.	0	[7:2], [31:16]	r	Reserved Read as 0; should be written with 0.
Field	Bits	Type	Description																					
0	0	rw	Reserved Read as 0; should be written with 0.																					
SWRSTREQ	1	w	Software Reset Request 0 _B No SW Reset is requested 1 _B A SW Reset request trigger is generated This bit is automatically cleared and read always as zero.																					
RES	[15:8]	rw	Reserved Should be written with original content.																					
0	[7:2], [31:16]	r	Reserved Read as 0; should be written with 0.																					
	<i>Id.</i> pg. 7-113 For example, in the alternative, the firmware may attempt to recover lockstep by, e.g. releasing the processors from the idle condition by writing 00 to the REQSLP field of the PMCSR_x register:																							

Claim 19	Exemplary Evidence of Infringement
	Exiting Idle mode In Idle mode, if there is no outstanding SMU Idle request to the CPU, the CPU will return to Run mode in response to the following events: • An interrupt / trap received from an interrupt / trap source mapped to the CPU. • An NMI trap request is received to wake-up all the CPUs. • A MSB bit wrap of the corresponding CPU Watchdog counter occurs. • Setting the register bits PMCSR_x.REQSLP = 00_B to set the CPU_x into Run mode. If there is an outstanding SMU Idle request to the CPU, then the CPU may only return to Run mode after an application reset or any higher reset. The system enters reset state on an Application, System reset or any higher reset. If it is woken by a watchdog timer overflow event routed via the SMU to the CPU or by an NMI or by an interrupt, the CPU will immediately vector to the appropriate interrupt / trap handler. <i>Id.</i> pg. 7-199 In the alternative, the firmware may attempt to recover lockstep operation by configuring the SMU to reset the processors on detection of loss of lockstep.  TC29x B-Step Safety Management Unit (SMU) – generate an Interrupt request to any of the CPUs, concurrent interrupts to several CPUs can be configured – generate a NMI request to the System Control Unit – generate a reset request to the System Control Unit leading to a system or application reset – activate the pad emergency stop signal controlling the safe state of output pads – generate a CPU idle request

Claim 19	Exemplary Evidence of Infringement
	<i>Id.</i> pg. 9-4



Claim 19	Exemplary Evidence of Infringement
	<i>Id.</i> pg. 9-2 The operating system may be informed that lockstep has been recovered in multiple ways, <i>e.g.</i>, detecting that the system has exited the reset condition. 7.2.1 Reset Operation This section describes the conditions under which the TC29x will be reset and the reset operation configuration and control. 7.2.1.1 Overview The following reset request triggers are available: • Supply monitor (SWD) triggers a power-on reset (cold reset) • 1.3V EVR monitor triggers a power-on reset (cold reset) • 3.3V EVR monitor triggers a power-on reset (cold reset) • Standby EVR (STBYR) monitor triggers a power-on reset (cold reset) • External active low hardware “power-on” reset request trigger; $\overline{\text{PORST}}$ (can be either a warm reset or to extend a cold reset) • External System Request reset trigger pins; $\overline{\text{ESR0}}$ and ESR1 (warm reset) • Safety Management Unit (SMU) alarm reset request trigger, (warm reset) • Software reset (SW), (warm reset) • System Timer (STMx) trigger (warm reset) • Resets via the JTAG interface • JTAG resets (special reset) • Software triggered module reset <i>Id.</i> pg. 7-96

Claim 19	Exemplary Evidence of Infringement																																																																																														
	7.2.1.13 Reset Controller Registers Status Registers After a chip level reset has been executed, the Reset Status registers provide information on the trigger of the last reset(s). Warm reset status bits are updated upon each reset cycle. A reset cycle is finished when all resets are de-asserted. Within a reset cycle the status flags are used as sticky flags. Module level resets have no effect on the Reset Status register. Bits which may indicate a cold reset trigger (SWD, EVR33, PORST and EVR13) provide useful information to the application and are not cleared automatically. The application may clear these bits by writing to bit RSTCON2.CLRC. RSTSTAT Reset Status Register (050_H)Reset Value: 1381 0000_H <table><tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr><tr><td colspan="3">0</td><td>STB YR</td><td>0</td><td>SWD</td><td>EVR 33</td><td>EVR 13</td><td colspan="3">0</td><td>CB3</td><td>CB1</td><td>CB0</td><td>0</td><td>POR ST</td></tr><tr><td colspan="3">r</td><td>rh</td><td>r</td><td>rh</td><td>rh</td><td>rh</td><td colspan="3">r</td><td>rh</td><td>rh</td><td>rh</td><td>r</td><td>rh</td></tr><tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr><tr><td colspan="7">0</td><td>STM 2</td><td>STM 1</td><td>STM 0</td><td>SW</td><td>SMU</td><td>0</td><td>ESR 1</td><td>ESR 0</td></tr><tr><td colspan="7">r</td><td>rh</td><td>rh</td><td>rh</td><td>rh</td><td>rh</td><td>r</td><td>rh</td><td>rh</td></tr></table>	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	0			STB YR	0	SWD	EVR 33	EVR 13	0			CB3	CB1	CB0	0	POR ST	r			rh	r	rh	rh	rh	r			rh	rh	rh	r	rh	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	0							STM 2	STM 1	STM 0	SW	SMU	0	ESR 1	ESR 0	r							rh	rh	rh	rh	rh	r	rh	rh
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16																																																																																
0			STB YR	0	SWD	EVR 33	EVR 13	0			CB3	CB1	CB0	0	POR ST																																																																																
r			rh	r	rh	rh	rh	r			rh	rh	rh	r	rh																																																																																
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0																																																																																
0							STM 2	STM 1	STM 0	SW	SMU	0	ESR 1	ESR 0																																																																																	
r							rh	rh	rh	rh	rh	r	rh	rh																																																																																	
	<i>Id.</i> pg. 7-105																																																																																														

Id. pg. 7-105

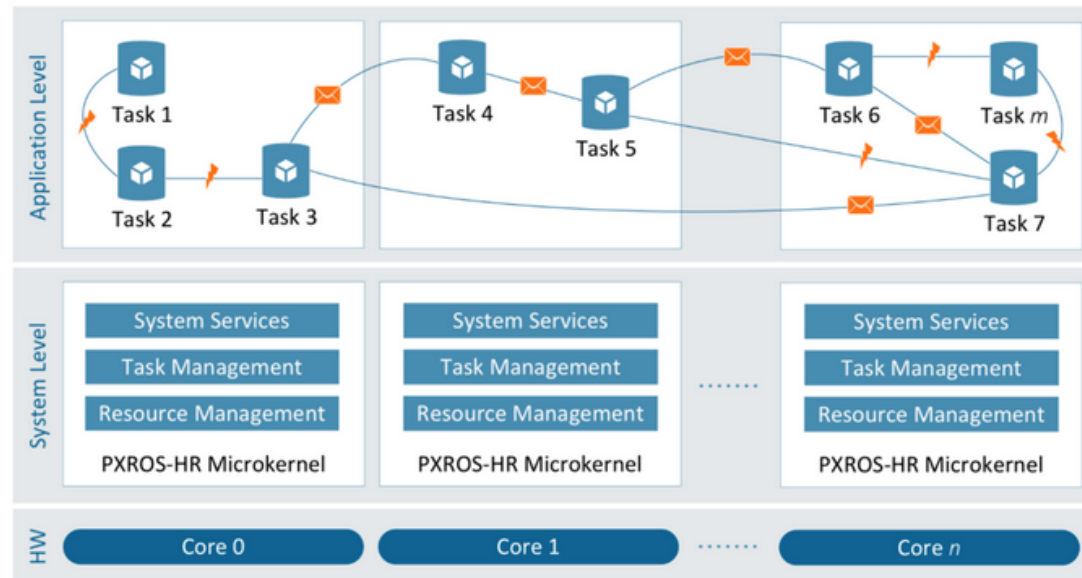
Claim 19	Exemplary Evidence of Infringement
	6.2 Lockstep Control The lockstep control function is enabled by the LSEN bitfield in a control register in the SCU. Each core capable of lockstep has its own instance of the control register. In this product, the CPU1 instance of the Tricore can be lockstepped so there is one register, LCLCON1 for CPU1. These registers are only initialised by a cold power-on reset. In this initialisation state, all lockstepped processors in the system will have lockstep enabled. The lockstep function can only be disabled by the system initialisation software writing a 0_B to the LSEN bitfield. Application software cannot enable or disable the lockstep function. The current mode of the lockstep logic can be monitored by reading the lockstep status bit, LS, in the associated LCLCON register. Writes to the control registers will be subject to the protection mechanisms of the SCU. <i>Id.</i> pg. 6-1 In a further example, the operating system may be informed that lockstep has been recovered via e.g., examining whether the SMU is still reporting a loss of lockstep.

Claim 19	Exemplary Evidence of Infringement								
	9.5.5SMU Alarm Status Registers SMU Alarm status registers related to the different alarm groups. SMU_AGx (x=0-6) Alarm Status Register(1C0_H + x*04_H)Reset Value: 0000 0000_H 31302928272625242322212019181716 SF31SF30SF29SF28SF27SF26SF25SF24SF23SF22SF21SF20SF19SF18SF17SF16 rwhrwhrwhrwhrwhrwhrwhrwhrwhrwhrwhrwhrwhrwhrwhrwh 1514131211109876543210 SF15SF14SF13SF12SF11SF10SF9SF8SF7SF6SF5SF4SF3SF2SF1SF0 rwhrwhrwhrwhrwhrwhrwhrwhrwhrwhrwhrwhrwhrwhrwhrwh <table><tr><th>Field</th><th>Bits</th><th>Type</th><th>Description</th></tr><tr><td>SFn (n = 0-31)</td><td>n</td><td>rwh</td><td> Status flag for alarm n belonging to alarm group x (x=0-6). 0_BStatus flag n does not report a fault condition 1_BStatus flag n reports a fault condition </td></tr></table> Id. pg. 9-115 In another alternative or in addition to the above, the firmware may inform the operating system of the recovery of lockstep. For example, on information and belief the Accused Products use an ASIL D (ISO26262) operating system such as the Hitex PXROS Real-time OS for TriCore and Aurix processors such as the TC29X.	Field	Bits	Type	Description	SFn (n = 0-31)	n	rwh	Status flag for alarm n belonging to alarm group x (x=0-6). 0_BStatus flag n does not report a fault condition 1_BStatus flag n reports a fault condition
Field	Bits	Type	Description						
SFn (n = 0-31)	n	rwh	Status flag for alarm n belonging to alarm group x (x=0-6). 0_BStatus flag n does not report a fault condition 1_BStatus flag n reports a fault condition						

Claim 19**Exemplary Evidence of Infringement**

PXROS - Real-time OS for TriCore™ and AURIX™

PXROS-HR is an object-oriented real-time operating system (RTOS) with a very modern microkernel. Its architecture makes it particularly suitable for use on modern multicore MCUs. The current version of PXROS-HR brings improvements in the concepts for encapsulation and has improved robustness. This is achieved by using hardware protection mechanisms that are part of modern microcontrollers such as Infineon's AURIX family.



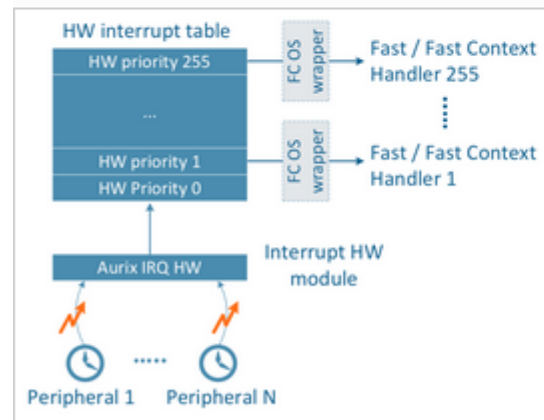
<https://www.hitex.com/tools-components/middleware/rtos/pxros-real-time-os-for-tricore-and-aurix>

Claim 19	Exemplary Evidence of Infringement
	For example, PXROS, like most modern Operating Systems allows low level code, such as interrupt handlers, to send messages to the operating system:

Tasks and Handlers

Because fast reactions to certain events are required in many technical applications, which should be executed with as little overhead as possible, PXROS implements the concept of complete interrupt transparency, i.e. PXROS never changes the state of the interrupt system of the microcontroller. This feature makes it possible to achieve warm start capability with PXROS. Thus, there are no interrupt locks caused by PXROS and interrupt services can always interrupt PXROS. The RTOS solely manages the remaining time resources of a main-loop. Due to the interrupt transparency, existing applications without OS can easily be ported to PXROS-HR as well as OS, which are partially or totally implemented on interrupt level. As the interrupt handling is not “predefined” or interfered by the microkernel, the application might implement any needed or reasonable check for the activity related to interrupts.

In the PXROS terminology the interrupt service routines are so called handlers, which, in principle, are completely under control of the application, but can still use a subset of PXROS services. These services are handled in a special way, in order to minimize the PXROS related overhead on interrupt level. A handler can, for example, send signals (Events) to a task. This service is not executed on interrupt level, but is inserted into a list instead. Then this service is executed before returning from the interrupt levels to task level. This is in time, because the next task to be executed has to be determined before leaving the interrupt level.



The use of handlers guarantees optimal time response (just like without OS), but except for the highest interrupt level, has effects on other interrupts. Actions within a task have no effect on the interrupt system, but task scheduling and communication generate some overhead, and thus it is vital for the overall performance of an application to find a good balance between handler and task based execution of application parts.

In PXROS philosophy, a handler always belongs to a task and its address space. This means that in PXROS-HR, handlers will be controlled by the MPU. Handlers and the related task have a relation like interrupts to main-loop.

In dynamically configured PXROS - systems (static configuration is also possible), tasks are responsible for installing and uninstalling related handlers. This concept allows dynamic configuration and reconfiguration of a running system.

Claim 19	Exemplary Evidence of Infringement
	https://hightec-rt.com/en/products/real-time-os.html

EXHIBIT C



Adaptive Cruise Control (MRR-20)

User Manual (KR ver.)



HL Klemove

I. 시스템 일반

1. 개요

- ACC(Adaptive Cruise Control) 시스템은 차량 전방에 장착된 레이더를 이용하여 선행 차량과의 거리 및 속도를 측정하여 선행 차량과 적절한 거리를 자동으로 유지하는 편의 기능이다. 또한 선행차량이 정차한 경우 선행차량을 따라 후방에 정차하며, 재 출발 시 함께 출발하는 기능인 Stop & Go 기능을 제공한다.

2. 시스템 동작



1) 일반제어

- 30 ~ 200 km/h 속도 범위에서, 설정 속도는 현재 자차의 속도로 설정된다.
- 0 km/h (정차 상태) 에서는 브레이크 페달을 밟은 상태로 작동 시킬 수 있다.
- 0 ~ 30 km/h 속도 범위에서는, 초기 설정 속도가 30 km/h 로 설정된다.

2) 정지제어

- 선행차량이 정차하면, 일정거리 뒤에 정차한다.
- 3초 이내 출발 시 자동 출발하며, 3초가 넘어가면 Resume / Set 스위치 조작 또는 가속 페달을 밟을 경우 출발한다. 5분 이상 정차유지 시 ACC 제어 해제된다.

* 차종/제조사/기능에 따라 시간은 바뀔 수 있음

HL Klemove

3) 오버라이드 제어

- 운전자가 가속 페달을 밟으면, 시스템의 판단보다 운전자의 의지를 우선하여 가속한다. 이후, 운전자가 가속 페달을 놓으면, 목표 속도로 서서히 감속한다.

4) 추월 지원 제어

- 운전자가 좌측 방향지시등을 조작하면, 시스템은 표시상태와는 관계없이 내부적으로 목표 거리를 1단계로 설정한다. 따라서 약간의 가속이 발생하여 추월을 부드럽게 할 수 있도록 도와준다.
- 60Kph 이상에서 작동하도록 되어 있다.

II. 시스템 구성 상세

1.ACC

1) 기능

- 전방 도로상황 모니터링 (센서 기능)
- 운전자 입출력 및 차량상태 모니터링 (ECU 기능)
- 요구 가속도 연산 및 출력 (ECU 기능)

2) 성능

- Radar 사양 별 상세 Spec. 은 Web Page 참조
(<https://www.hlklemove.com/business/AD-solution/radar/front.do>)

2. 스위치

1) 기능

- ACC 시스템 On / Off
- 목표 속도 조작
- 목표 거리 조작
- 동작 진입 / 해제
- ACC ↔ CC 전환 조작

* 차량/제조사에 따라 다를 수 있음



HL Klemove

3. 클러스터

1) 기능

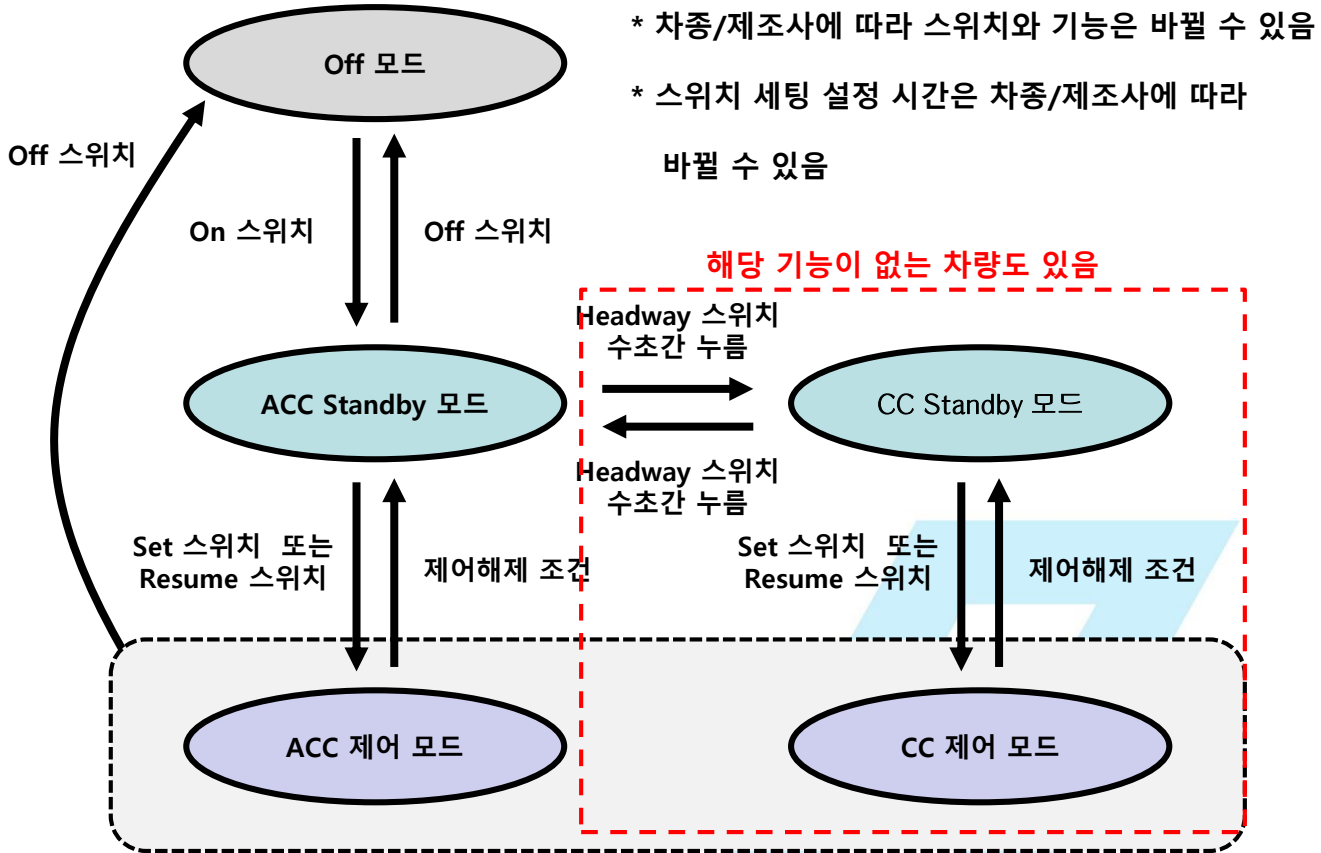
- 시스템 On / Off
- 시스템 경고등
- ACC 동작상태(선행차량, 제동, 오버라이드, 정차모드) 표시
- ACC 운전자 설정상태(목표속도, 목표거리) 표시
- ACC 자동해제 / 사용조건 불만족
- ACC 위험상황(선행차량 급제동, 정지차량 주의) 표시



HL Klemove

III. 시스템 기능 상세

2. ACC 동작모드



- 1) 시동을 걸면 Off 모드
- 2) On 스위치를 누르면 ACC Standby 모드
 - "CRUISE" 램프 점등
- 3) 주행 중, Set 스위치를 누르면 ACC 제어모드 (ACC 기능 동작)
- 4) 브레이크를 밟으면 ACC Standby 모드 (ACC 기능 해제)
- 5) Off 스위치를 누르면 Off모드 (어떤 모드에서도 Off 조작 시, Off모드로 이동)
 - "CRUISE" 램프 소등
- 6) 차간거리 스위치를 수초간 누르면 CC Standby 모드 (해당 기능이 없는 차량도 있음)
 - 주행 중, Set 스위치를 누르면 CC 제어모드 (CC 기능 동작)
 - 브레이크를 밟으면 CC Standby 모드 (CC 기능 해제)

- 7) 차간거리 또는 Mode 스위치를 수초간 누르면 ACC Standby 모드

HL Klemove

3. 스위치를 이용한 ACC 조작법

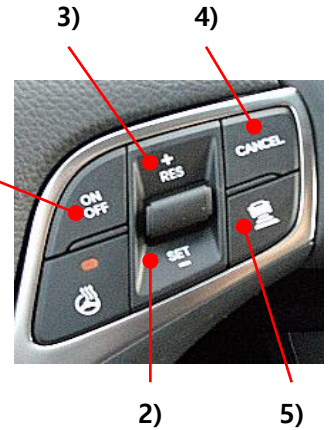
* 차종/제조사에 따라 스위치는 바뀔 수 있음

1) 『ON / OFF』 메인 스위치

- 시스템 ON, OFF 기능을 수행한다.
- Standby 모드로 전환 시, "CRUISE" 램프 점등

2) 『SET/-』스위치

- Standby 모드에서 스위치를 누르면, 제어를 시작한다.
 - » 0 ~ 30 km/h 구간에서는 목표속도가 30 km/h 로 설정
 - » 30 km/h 이상에서는 현재속도가 목표속도로 설정
- 제어 중 스위치를 누르면, 목표속도가 감소한다.
 - » 짧게 누르면 1km/h씩 감소. 예) 81→80→79→78
 - » 길게 누르면 처음 1km/h 감소 후, 10 km/h 단위로 감소. 예) 77→76→70→60



3) 『RES/+』 스위치

- Standby 모드에서 스위치를 누르면 제어를 시작한다.
 - » Off → Standby 모드 전환 후, 한번이라도 ACC 동작시킨 기록이 있어야 한다.
- 제어 중, 스위치를 누르면, 목표속도가 증가한다.
 - » 짧게 누르면 1km/h씩 증가. 예) 77→78→79→80→81
 - » 길게 누르면 처음 1km/h 증가 후, 10 km/h 단위로 증가. 예) 66→67→70→80

4) 『CANCEL』 스위치

- 제어상태에서 해제된다.

5) 『거리설정』 스위치

- 목표차간거리를 설정한다(설정 초기값은 4단).
- 스위치를 누르면 목표차간거리가 다음 순서대로 변경된다.
 - » 4단 → 3단 → 2단 → 1단 → 4단
- 차량 속도가 90km/h (25m/s)일 경우,
 - » Level1 (1.0초)일 때, Headway distance는 25m.
 - » Level2 (1.3초)일 때, Headway distance는 32.5m
 - » Level3 (1.6초)일 때, Headway distance는 40m
 - » Level4 (2.1초)일 때, Headway distance는 52.5m

• Standby 모드에서 2초 이상 누르면, ACC와 CC 모드의 전환을 한다.

HL Klemove

4. 클러스터 표시사항의 이해

1) ACC 표시 탭의 적용

- DH 의 클러스터에는 ACC 의 동작상태를 표시하는 별도의 탭이 존재한다. 해당 탭에서 SCC 관련 정보를 확인할 수 있다.

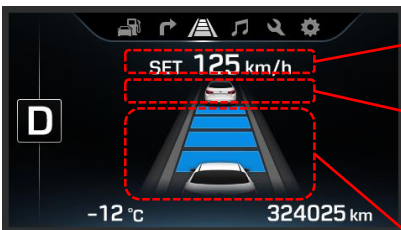


* 차종/제조사에 따라 스위치는 바뀔 수 있음

2) 메인 스위치 On 조작 시, Cruise 램프 점등



3) 주행 중, Set / Resume 스위치를 조작하여 ACC 제어를 시작하면, ACC 동작상태가 표현



▶ 목표 속도 설정 값에 따라 (오버라이드 시, 점멸)

SET 125 km/h

▶ 선행차량 유무에 따라



▶ 목표 차간거리 설정 값에 따라



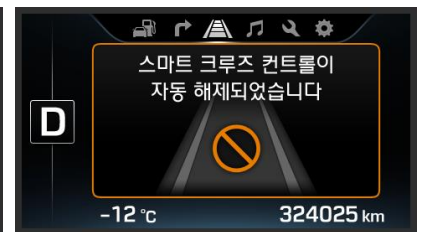
4) 선행차량 뒤에 정차하여 정차상태 유지할 경우



▶ 정차 후, 3초 이전



▶ 정차 후, 3초 이후



▶ 정차 후, 5분 이후

* 차종/제조사에 따라 시간은 바뀔 수 있음 7



Adaptive Cruise Control (MRR-20)

User Manual (EN ver.)





I. System Information

1. Introduction

- The ACC system operates the convenience feature automatically maintaining the distance by measuring the distance and speed of the lead vehicle with the radar in front of the car. Also, if the lead vehicle is stopped along the rear of the lead vehicle stops, and re-start feature provides at the start with Stop & Go function.

2. System Operation

operation principle

Absence of a lead vehicle

Existing of a lead vehicle

Stopping a lead vehicle

Blinding a lead vehicle

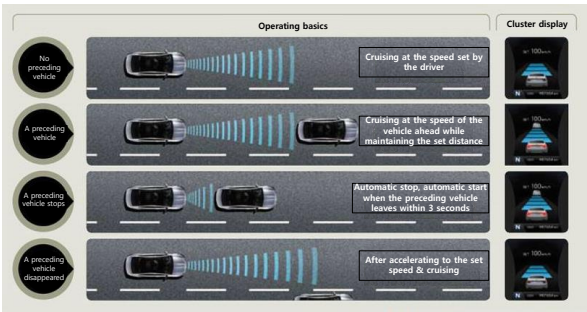
Cluster display state

Cruise control at the driver's setting speed

Driving the lead vehicle traveling speed maintaining a set following distance

When auto stop and starting the lead vehicle within 3 seconds, automatically start.

Accelerate to the target speed and then cruise control.



1) Normal Mode Control

- All of the speed/distance control can perform from 30 to 200 km/h speed range.
- The minimum target speed is 30km/h.
- If the lead vehicle's speed is from 0 to 30km/h, the proper distance can be expected(or calculated/measured).

2) Stop mode Control

- When the lead vehicle is stopped, and then the vehicle would stop at a certain distance.
- [HDA not operated / HDA during operated] If the stop duration is within 3 sec. / 30 sec., the vehicle automatically start, or not manually the resume/set switch turn on or the accelerator pedal is pressed. If the stopping maintain more than 5 minutes, the ACC control will be released.

The control time be adjustable by OEM/Chassis/Funtion.

3) Overdrive Control

- Driver's accelerator pedal is pressed, the system will give acceleration priority to the judgment of the driver than the system decision. Then, the driver releasing the accelerator pedal, the vehicle speed gradually will going target speed.

4) Overtaking Support Control

- The driver turn on the left blinker, the system internally set as the target distance automatically-step regardless of the display condition. Therefore, a slight acceleration can help to occur soft overtaking.
- The function can be operated 60kph over.

HL Klemove

II. Specific System Structure

1. ACC

1) Function

- Monitoring front road condition(sensing function)
- Driver riding/getting off and monitoring car condition(ECU function)
- Calculating and output of the Required acceleration(ECU function)

2. Switch

1) Function

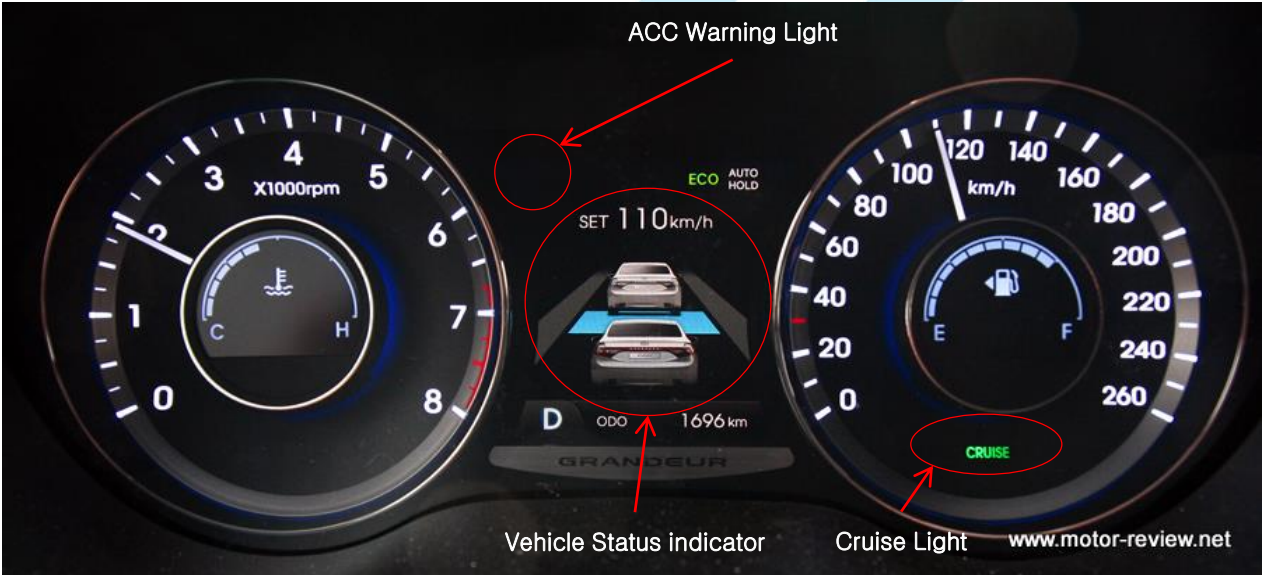
- ACSS system On/Off
- Target speed control
- Target distance control
- Operation ON/OFF
- ACC-CC switching operation



3. Cluster

1) Function

- System On/Off
- System Alarm
- ACC Operation State (lead vehicle, Break, Overtake, Stop modes) Display
- ACC Driver Setting State (Target Speed/Distance) Display
- ACC Auto-unlock/ Usage Condition Un-satisfaction
- ACC Hazardous situation (lead vehicle breaking, Stop vehicle alert) Display



HL Klemove

4. Instruction of the ACC control using switch

1) ON/OFF Main Switch

- ON/OFF function
- In case of the Standby mode, "CRUISE" Ramp turn ON

2) 『SET/-』 Switch

- At the Stand by mode, Press SET Switch and then Control start.
 - » The target speed set 30 km/h when the car speed is from 0 to 30 km/h.
 - » When the car speed is over 30 km/h, the target speed is the same as current speed.
- If the SET switch is pressed under control, the target speed will be decreased.
 - » Press short period, the speed decrease 1km/h step. ex) 81→80→79→78
 - » Press long period, the speed decrease at first 1km/h step then 10km/h step. ex) 77→76→70→60

3) 『RES/+』 Switch

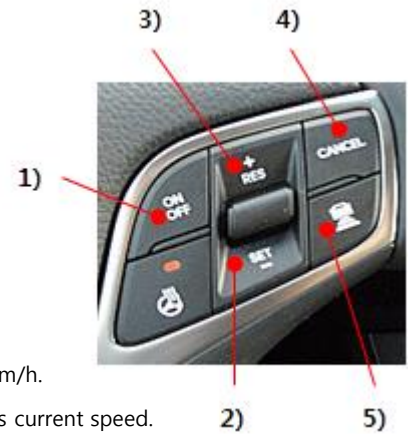
- At Standby mode, Press RES+ switch and then control start.
 - » After the OFF mode changed to Standby mode, the ACC usage history have to be existed at least once.
- If the RES+ switch is pressed under control, the target speed will be increased.
 - » Press short period, the speed increase 1km/h step. ex) 77→78→79→80→81
 - » Press long period, the speed increase at first 1km/h step then 10km/h step. ex) 66→67→70→80

4) 『CANCEL』 Switch

- Released from the control condition

5) 『Distance setting』 switch

- Set the target headway distance(Initial value is level.4)
- Press the switch, the target headway distance value changed in the following order.
 - » level.4->level.3->level.2->level.1->level.4
 - » In case of the car speed is 90km/h
 - » At level 1 (1sec.), Headway distance is 25m.
 - » At level 2 (1.3sec.), Headway distance is 32.5m
 - » At level 3 (1.6sec.), Headway distance is 40m
 - » At level 4 (2.1sec.), Headway distance is 52.5m
- If press the switch over 2 sec, the ACC mode is switched the CC mode each other.





Notice:

This Device complies with Part 15 of the FCC Rules [and with Industry Canada licence-exempt RSS standard(s)].

Operation is subject to the following two conditions:

- (1) This device may not cause harmful interference, and*
- (2) This device must accept any interference received, including interference that may cause undesired operation.*

Le présent appareil est conforme aux CNR d'Industrie Canada applicables aux appareils radio exempts de licence. L'exploitation est autorisée aux deux conditions suivantes:

- (1) l'appareil ne doit pas produire de brouillage, et*
- (2) This device must accept any interference received, including interference that may cause undesired operation.*

Notice:

Changes or modifications made to this equipment not expressly approved by (manufacturer name) may void the FCC authorization to operate this equipment.





Adaptive Cruise Control (MRR-20)

User Manual (CN ver.)



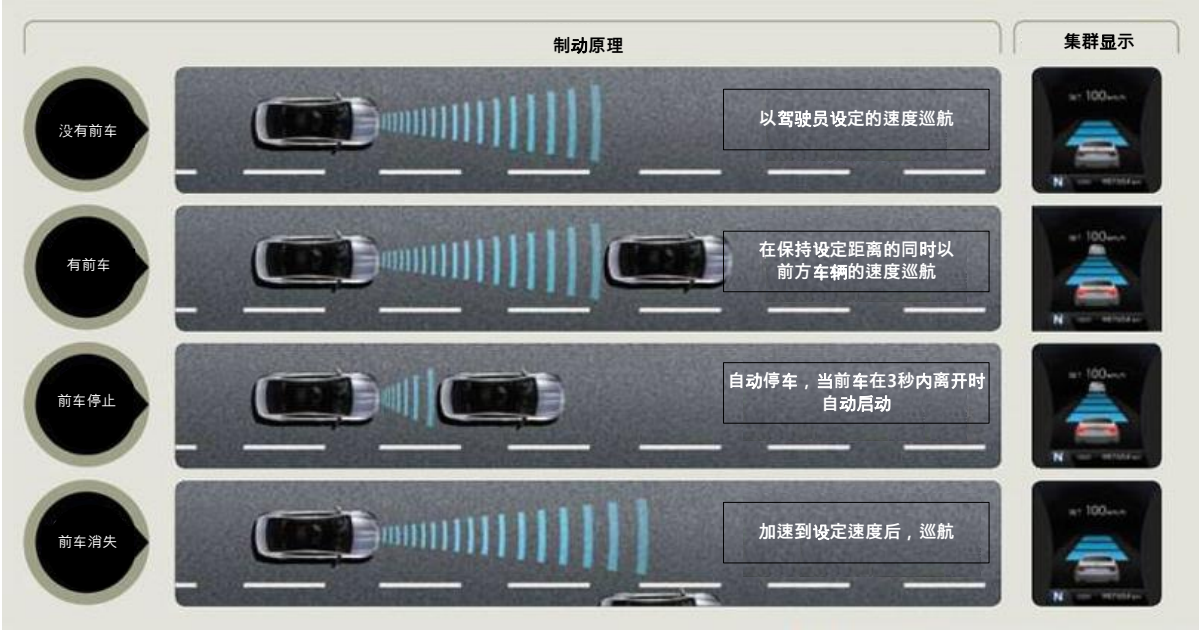


I. 系统信息

1. 介绍

- ACC(Adaptive Cruse Control)系统是利用安装在车辆前方的雷达测定与先行车辆的距离及速度,自动维持与先行车辆适当距离的功能. 另外,在先行车辆停靠的情况下,将跟随先行车辆停靠后方,提供重新出发时一起出发的功能Stop & Go.

2. 系统动作



1) 普通控制

- 在0~200km/h速度范围内, 设定速度以当前自车的速度设定.
- 在0km/h(车辆静止)时, 可以在踩刹车踏板的状态下启动
- 在0~30km/h速度范围内, 初始设定速度设定为30km/h

2) 静止控制

- 先行车辆停车后, 在一定距离后面停车
- 3秒以内出发时自动出发, 超过3秒后, 操作Resume/Set开关或踩油门踏板时出发, 停车维持5分钟以上时, ACC控制将被解除

3) 超驰控制

- 当驾驶员踩下油门踏板时, 比起系统, 更优先判断驾驶员的意志并加速. 之后, 驾驶员松开油门踏板, 就会慢慢减速到目标速度.

4) 超越支持控制

- 当驾驶员操作左侧方向指示灯时, 系统将与显示状态无关, 在内部将目标距离设置为第一阶段, 因此, 稍微加速有助于顺利超越

II. 系统构成详细

1.SCC

1) 功能

- 前方路况监测 (传感器功能)
- 驾驶员输入/输出及车辆状态监控 (ECU功能)
- 请求加速运算及输出 (ECU功能)

2) 性能

- 规格。 请参阅网页。 .
(<https://www.hlklemove.com/business/AD-solution/radar/front.do>)

2. SWITCH

1) 功能

- ACC 系统 On / Off
- 目标速度操作
- 目标距离操作
- 动作进入/解除
- ACC ↔ CC 转换操作



HL Klemove

3. CLUSTER

1) 功能

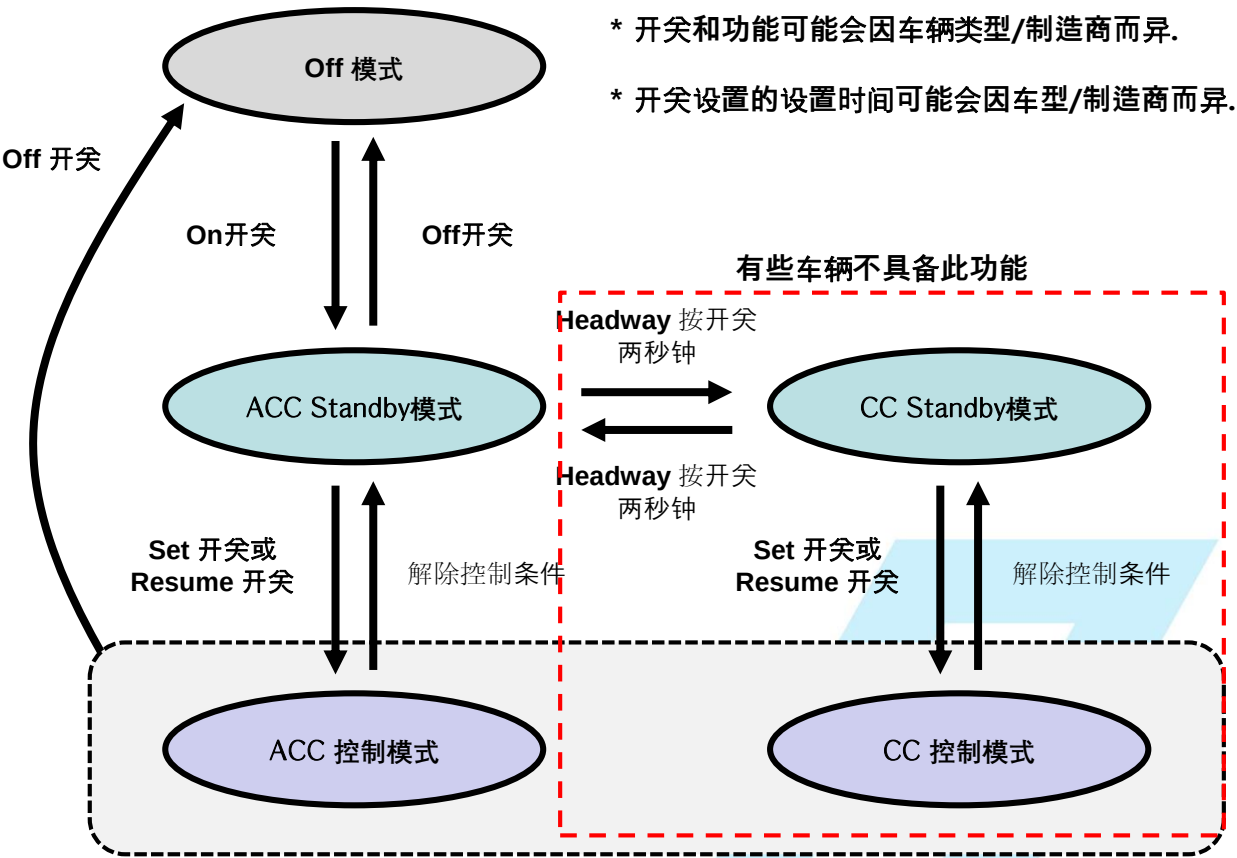
- 系统 On / Off
- 系统警示灯
- 显示 ACC 工作状态(先行车辆, 制动, 超控, 停车模式)
- 显示 ACC 驾驶员设定状态(目标速度, 目标距离)
- ACC自动解除 / 不满足使用条件
- 显示 ACC 危险情况 (前方车辆紧急刹车, 注意静止车辆)



HL Klemove

III. 系统功能详解

2. ACC 动作模式



- 1) 启动后Off模式
- 2) 按下On开关,ACC Standby模式
 - "CRUISE"大灯亮起
- 3) 行驶中,按下Set开关,ACC控制模式(ACC功能动作)
- 4) 刹车后ACC Standby模式(ACC功能解除)
- 5) 按下Off开关,Off模式(无论在何种模式下操作Off时,移动至Off模式)
 - "CRUISE"大灯熄灭
- 6) 按一下车距开关数秒钟,CC Standby模式 (某些车辆不具有此功能)
- 7) 行驶中,按下Set开关,CC控制模式(CC功能动作)
- 8) 踩刹车的话CC Standby模式(CC功能解除)
- 9) 按下车距或模式开关几秒钟,ACC Standby模式

HL Klemove

3. 使用开关的ACC操作方法

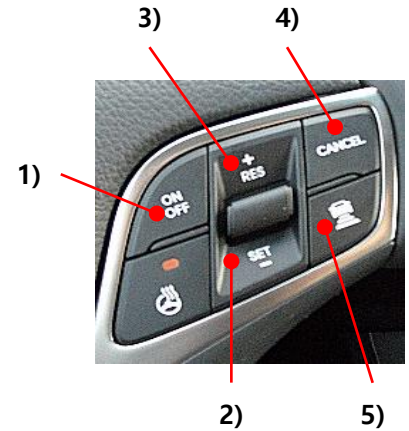
* 开关可能会因车型/制造商而异

1) 『ON / OFF』 主开关

- 执行系统ON,OFF功能。
- 转换为Standby模式时,“CRUISE”灯亮起

2) 『SET/-』 开关

- 在Standby模式下按下开关, 就开始控制.
 - » 0 ~ 30 km/h 区间内设定目标速度为 30 km/h
 - » 在30km/h以上时,将当前的速度设定为目标速度
- 在控制中按下开关, 目标速度就会减慢.
 - » 短按可减少1km/h. ex) 81→80→79→78
 - » 长按时, 首次减少1km/h后, 以10km/h为单位减少. ex) 77→76→70→60



3) 『RES/+』 开关

- 在Standby模式下按下开关, 就开始控制.
 - » Off → Standby模式切换后, 至少应有一次ACC动作记录.
- 在控制中,按下开关, 目标速度就会增加.
 - » 短按可增加1km/h. ex) 77→78→79→80→81
 - » 按压长,第一次增加1km/h后,以10km/h为单位增加. ex) 66→67→70→80

4) 『CANCEL』 开关

- 解除控制状态.

5) 『距离设定』 开关

- 设定目标间隔距离(设定初始值见第4段).
- 按下电门, 目标差间距按下一顺序变更.
 - » 4当 → 3当 → 2当 → 1当 → 4当
- 车速为90km/h (25m/s)时,
 - » 在Level1 (1.0秒)时,Headway distance为25m
 - » 在Level2 (1.3秒)时, Headway distance为32.5m
 - » 在Level3 (1.6秒)时, Headway distance为40m
 - » 在Level4 (2.1秒)时, Headway distance为52.5m
- 在Standby模式下按压2秒以上,即可转换AACC和CC模式.

HL Klemove

4. 理解CLUSTER显示

1) ACC 标靶的适用

- DH的集群中存在显示ACC运转状态的特别插销. 可以在相应程序中确认SCC相关信息.

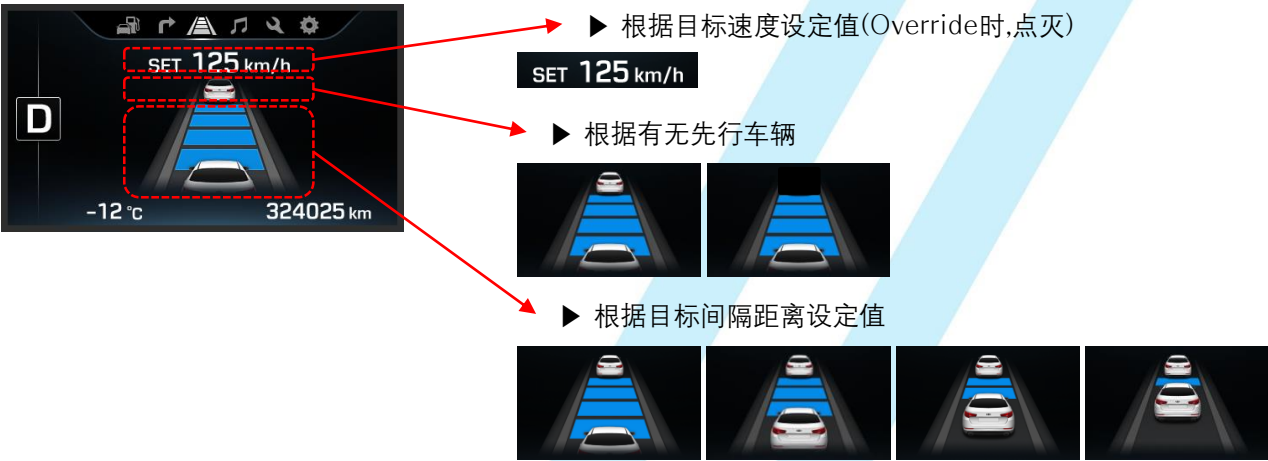


* 开关可能会因车辆型号/制造商而异

2) 主开关On操作时,Cruise灯点灯



3) 行驶中,操作Set / Resume开关开始控制ACC,即可显示ACC动作状态。



4) 先行车辆后面停车维持停车状态时



* 时间可能会因车型/制造商而异



대한민국 , Republic of Korea



기자재의 명칭 : 특정 소출력 무선기기
(차량 충돌방지용 레이더 무선기기)
모델명 : MRR-20
인증번호 : R-CMM-MF3-MRR-20
상호 : 주식회사 에이치엘클레무브
제조년월일 : 제품 별도 표기
제조자 : 주식회사 에이치엘클레무브
제조국 : 대한민국

United States of America



FCC ID
: 2A30Z-MRR20
This device complies with Part 15 of the FCC Rules. Operation is subject to the following two conditions:
(1) this device may not cause harmful interference, and
(2) this device must accept any interference received, including interference that may cause undesired operation.

CAUTION TO USERS
Changes or modifications not expressly approved by the party responsible for compliance could void the user's authority to operate the equipment.

מדינת ישראל, Israel

Ministry of Communication permit number : 55-11767

א. השימוש במכשיר פטור מרשיון הפעלה אלחוטית
לא מוגן מהפרעות וללא הפרעה למערכות אחרות הפועלות כדין.
ב. רק "בפעולת בזק" לשימוש עצמי של הלקוח בלבד, הצידוד פטור מרשיון הפעלה אלחוטי.
מתן "שרות בזק" לצד ג' מחייב רשיון מיוחד ממשרד התקשורת.
ג. אסור להחליף את האנטנה המקורית של המכשיר, ולא לעשות בו כל שינוי טכני אחר.



CANADA

Model: MRR-20
IC: 27992-MRR20

This device complies with Industry Canada licence-exempt RSS standard(s).
Operation is subject to the following two conditions:

- (1) this device may not cause interference, and
- (2) this device must accept any interference,
including interference that may cause undesired operation of the device.

Le présent appareil est conforme aux CNR. d'Industrie Canada applicables aux
appareils radio exempts de licence. L'exploitation est autorisée aux deux
conditions suivantes:

- (1) l'appareil ne doit pas produire de brouillage, et
- (2) l'utilisateur de l'appareil doit accepter tout
brouillage radioélectrique subi, même si le brouillage est susceptible d'en
compromettre le fonctionnement.

European Union



Model : MRR-20

Hereby MRR-20 has been so constructed that it
can be operated in at least one Member State
without infringing applicable requirements of
use of radio spectrum. (RED article 10.2)

Hereby, HL Klemove Corp. declares that the
radio equipment type MRR-20 is in compliance
with Directive 2014/53/EU.

The full text of the EU declaration of conformity
is available at the following as next page.

Website link
<https://hlklemove.com/solutions.html>



Russian Federation



新加坡共和国, Republic of Singapore

Complies with
IMDA Standards
N1588-22

United of Kingdom



Commonwealth of Australia



HL Klemove

Republika Srbija, Republic of Serbia



سلطنة عُمان, Sultanate of Oman

OMAN-TRA

TRA/TA-R/13399/22

D202960

المغرب, Kingdom of Morocco

Numéro d'agrément : MR 00031999 ANRT 2022
Date d'agrément : 2022-02-24

Estados Unidos Mexicanos, United Mexican States

IFETEL : RCPHLMR22-2324

"La operación de este equipo está sujeta a las siguientes dos condiciones:
(1) es posible que este equipo o dispositivo no cause interferencia perjudicial
(2) este equipo o dispositivo debe aceptar cualquier interferencia, incluyendo la que pueda causar su operación no deseada."
and RCPHLMR22-2324

HL Klemove

الإمارات العربية المتحدة ,The United Arab Emirates (UAE)



TDRA - UNITED ARAB Emirates
Mode Dealer ID Name: 5M GLOBAL LLC
TARTTE: ER05598/22
Model Name: MRR-20
Product Type: Vehicle Radar



República del Paraguay



NR 2021-03-I-0134

República Federativa do Brasil, Federative Republic of Brazil



03849-18-09644

"Este equipamento não tem direito à proteção contra interferência prejudicial e não pode causar interferência em sistemas devidamente autorizados."

Para maiores informações, consulte o site da ANATEL – www.anatel.gov.br

República Argentina



+ H-28083

HL Klemove

Republic of Indonesia



83759/SDPPI/2022
13085



Dilarang melakukan perubahan spesifikasi yang dapat menimbulkan gangguan fisik dan/atau elektromagnetik terhadap lingkungan sekitarnya

المملكة الأردنية الهاشمية , Hashemite Kingdom of Jordan

TRC/31/9732/2022

Republic of South Africa



TA-2021/3112
Approved

Republika ng Pilipinas, Republic of the Philippines



Type Approved
No. ESD-RCE-2229458

HL Klemove

O‘zbekiston Respublikasi, Republic of Uzbekistan



ราชอาณาจักรไทย, Thai



Ukraine



UA RF : 3HLKL20MR

Malaysia





TAIWAN / Republic of China



CCAO22LP1180T9

低功率電波輻射性電機管理辦法

第十二條 經型式認證合格之低功率射頻電機，非經許可，公司、商號或使用者均不得擅自變更頻率、加大功率或變更原設計之特性及功能。

第十四條 低功率射頻電機之使用不得影響飛航安全及干擾合法通信；經發現有干擾現象時，應立即停用，並改善至無干擾時方得繼續使用。

現象時，應立即停用，並改善至無干擾時方得繼續使用。

前項合法通信，指依電信法規定作業之無線電通信。

低功率射頻電機須忍受合法通信或工業、科學及醫療用電波輻射性電機設備之干擾。

使用此產品時應避免影響附近雷達系統之操作



中华人民共和国, People's Republic of China

车辆驾驶辅助雷达系统型号：MRR-20
执行标准：汽车雷达无线电管理暂行规定 2021 181号文
频率范围：76-77 GHz
放射功率：等效全向辐射功率(EIRP) $\leq 50\text{dBm}$
天线类型：印刷阵列天线
用户控制：不可
使用温度： $-40^{\circ}\text{C} \sim +85^{\circ}\text{C}$
电压：DC 12.0V
CMIIT ID：2022LJ14436

不得擅自更改发射频率、加发射功率（包括额外加装射频功率放大器），
不得擅自外接天线或改用其它发射天线

使用时不得对各种合法的无线电通信业务产生有害干扰；一旦发现有干扰现象时，
应立即停止使用，并采取措施消除干扰后方可继续使用

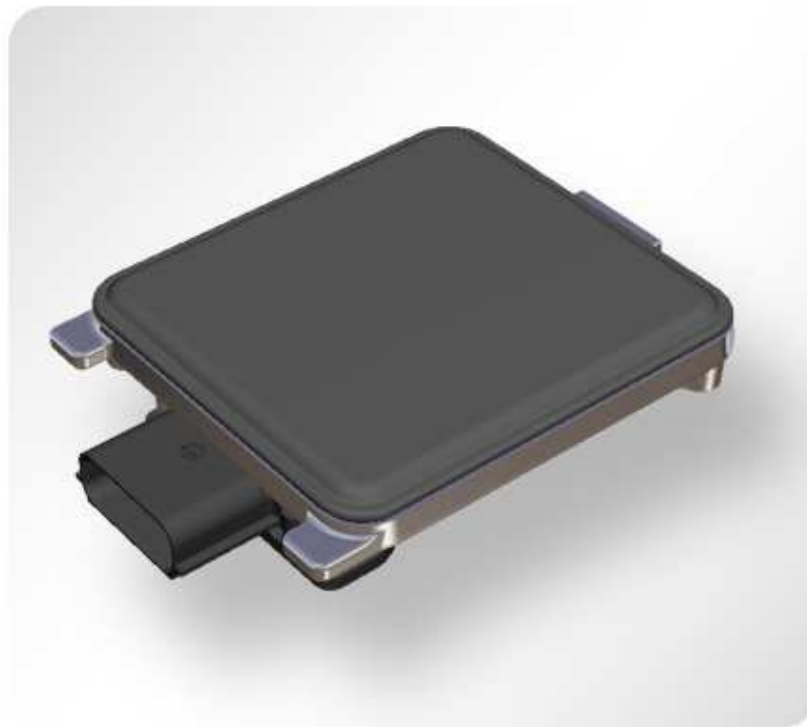
使用微功率无线电设备，必须耐受各种无线电业务的干扰或工业、科学及医疗应用设备
的辐射干扰

机场等的电磁环境保护区域内使用微功率设备，应当遵守电磁环境保护及相关行业
主管部门的规定

EXHIBIT D

 HELLA Hella GmbH & Co. KGaA 59552 Lippstadt	Date: 2021-12-08 No.: Page 1
<i>User's Guide</i>	
Subject: RS 5.4 – Advanced driver assistance system – User's Guide Ref.:	

RS 5.4 – User's Guide



The RS 5.4 is an advanced driver assistant system, to warn the driver of the ego vehicle against potential collisions with other road users to the side, to the rear and to the front of the ego vehicle.

This system is not meant to encourage aggressive driving. The absence of a warning will not guarantee the absence of other road users. Responsibility for the safe operation of the vehicle remains with the driver.

signed by:	checked by:
------------	-------------

 Hella GmbH & Co. KGaA 59552 Lippstadt	Subject: RS 5.4 – Advanced driver assistance system – User's Guide	Date: 2021-12-08
		No.:
		Page 2 of 8

1 Functions

The sensor is capable to detect moving object. With this recognition capability it covers the following functionality:

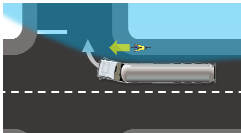
1.1 LCW



LCW (Lane Change Warning) is the feature used for recognition of target vehicles in the blind spot zone or in further distance in order to inform the driver of the ego vehicle about the safety level of a lane change. In case that a target vehicle approaches the ego vehicle in that way, that the lane change is no longer safe, a warning lamp is activated. If the driver of the ego vehicle decides to change the lane in such a dangerous situation, a warning can be given by setting the warning LED to blink.

Other HELLA terms are BSD = Blind Spot Detection and LCA = Lane Change Assistant

1.2 CTA



The City Turn Assist is the function supporting the truck driver during turning maneuvers and to keep an overview of blind spot zone of the truck.

In HELLA terminology there is a risk of mixing the abbreviation "CTA" with the Cross Traffic Alert-function, which is referring to rear cross traffic warnings. Therefore the City Turn Assist is often abbreviated as "TA".

1.3 FTA



The Front Traffic Alert (also known as Front Cross Traffic Alert (FCTA)) supports the driver on traffic in front of the vehicle, which is especially helpful on the driver diverted side.

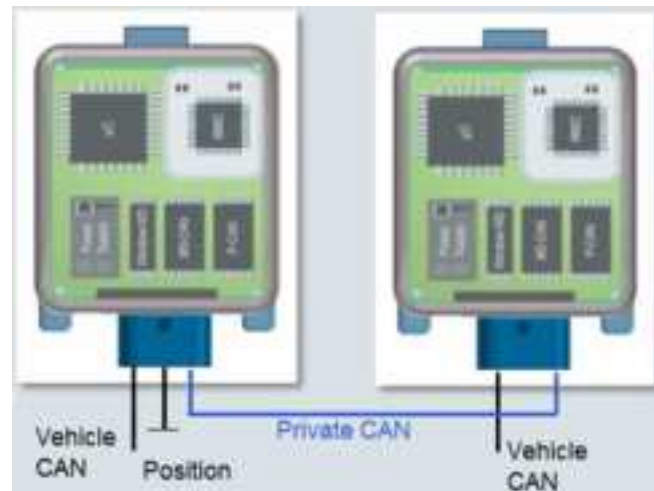
This document is confidential. Its contents are not to be exploited, passed on or disclosed to third parties without our express permission. All rights are reserved.

 Hella GmbH & Co. KGaA 59552 Lippstadt	Subject: RS 5.4 – Advanced driver assistance system – User's Guide	Date: 2021-12-08
		No.:
		Page 3 of 8

2 System Architecture

The system consists of two or more radar sensor units which are mounted in the front or rear corners of a vehicle or on the left and right sides of a vehicle so that the respective peripheral sectors of the vehicle can be observed.

The units interchange data between each other and the vehicle via the sensor CAN-bus.
All units incorporate an RF part and a microcontroller to perform the radar signal processing.



Block diagram of example system architecture.

This document is confidential. Its contents are not to be exploited, passed on or disclosed to third parties without our express permission. All rights are reserved.



Subject:

RS 5.4 – Advanced driver assistance system – User's Guide

Date: 2021-12-08

No.:

Page 5 of 8

Technical Data

Supply voltage	+9 V ... +32 V
Power consumption	Typ. 4 W
Operating frequency range	76000 MHz ... 77000 MHz
Modulation bandwidth	< 1000 MHz
Modulation	FMCW (fast chirps)
Antenna feed power	< 10 dBmW average
Antenna type	Microstrip patch array
Transmit antenna gain	14 dBi
Operating temperature range	-40°C ... +85°C

Abbreviations

ADC	Analog-to-digital converter
BPF	Band pass filter
FMCW	Frequency modulated continuous wave
IF	Intermediate frequency
PCAN	Private controller area network
RF	Radio frequency
Rx	Receive
Tx	Transmit
VCAN	Vehicle controller area network
VCO	Voltage controlled oscillator
WD	Watch dog

This document is confidential. Its contents are not to be exploited, passed on or disclosed to third parties without our express permission. All rights are reserved.

Signed by:

Checked by:

 Hella GmbH & Co. KGaA 59552 Lippstadt	Subject: RS 5.4 – Advanced driver assistance system – User's Guide	Date: 2021-12-08
		No.:
		Page 6 of 8

3 Vehicle Integration

Due to the fact that RADAR waves can penetrate plastics, the sensor integration is possible behind the bumper fascia and thus invisible from the exterior. However, the plastic and other materials which surround the sensor may cause bending, refraction and reflection of the RADAR waves. Distances, clearances, selected radii and other constructive elements in their arrangement can lead to constructive or destructive interference of the RADAR waves. In case of an unfavourable configuration of all parameters towards each other, the sensor properties can be influenced disadvantageously. For these reasons, a generally valid positioning guideline covering all possible configurations and degrees of freedom and guaranteeing undisturbed sensor properties cannot be given.

The sensors should be positioned in the vehicle at a height of 400 to 700 mm above the road surface. If the sensor is positioned too close above the road surface, increased road reflections may appear. Thus, an installation height as high as possible must be aimed for.

If the height specification cannot be fulfilled, then the planned installation height must be evaluated and agreed upon with Hella GmbH. & Co. KGaA. The amount of road reflections can for example be reduced by inserting a special additional shield below the radar sensor.

Spatial orientation of the sensors

The RS5.4 sensors can be mounted in the front, lateral and/or rear area of a vehicle under defined angles.

Examples for typical integration:

Lateral/ side integration: 85...95 deg



Signed by:

Checked by:

 Hella GmbH & Co. KGaA 59552 Lippstadt	Subject: RS 5.4 – Advanced driver assistance system – User's Guide	Date: 2021-12-08
		No.:
		Page 7 of 8

Rear side integration: 45 deg (depending on required field of view)



Front side integration: 45 deg (depending on required field of view)



This document is confidential. Its contents are not to be exploited, passed on or disclosed to third parties without our express permission. All rights are reserved.

Hella 3399DE (2000-05)

Signed by: _____

Checked by: _____

 Hella GmbH & Co. KGaA 59552 Lippstadt	Subject: RS 5.4 – Advanced driver assistance system – User's Guide	Date: 2021-12-08
		No.:
		Page 8 of 8

4 Legal text requirements

Note: This User's Guide is not intended for the end user. The RS5.4 Advanced Driver Assistance System is not sold separately from the vehicle. All the full legal texts must be provided by the OEM in the User's Manual of the vehicle.

USA:

Changes or modifications not expressly approved by the party responsible for compliance could void the user's authority to operate the equipment.

This device complies with part 15 of the FCC Rules. Operation is subject to the following two conditions: (1) This device may not cause harmful interference, and (2) this device must accept any interference received, including interference that may cause undesired operation.

Radio frequency radiation exposure Information: This equipment complies with FCC radiation exposure limits set forth for an uncontrolled environment. This equipment should be installed and operated with minimum distance of 20 cm between the radiator and your body. This transmitter must not be co-located or operating in conjunction with any other antenna or transmitter.

Canada (both English and French language required):

This device contains licence-exempt transmitter(s)/receiver(s) that comply with Innovation, Science and Economic Development Canada's licence-exempt RSS(s). Operation is subject to the following two conditions:

- (1) This device may not cause interference.
- (2) This device must accept any interference, including interference that may cause undesired operation of the device.;

Le présent appareil est conforme aux CNR d'Industrie Canada applicables aux appareils radio exempts de licence. L'exploitation est autorisée aux deux conditions suivantes:

- (1) l'appareil ne doit pas produire de brouillage, et (2) l'utilisateur de l'appareil doit accepter tout brouillage radioélectrique subi, même si le brouillage est susceptible d'en compromettre le fonctionnement.

This equipment complies with ISSED radiation exposure limits set forth for an uncontrolled environment. This equipment should be installed and operated with minimum distance of 20 cm between the radiator and your body. This transmitter must not be co-located or operating in conjunction with any other antenna or transmitter.

Cet équipement est conforme aux limites d'exposition aux rayonnements ISSED établies pour un environnement non contrôlé. Cet équipement doit être installé et utilisé avec un minimum de 20 cm de distance entre la source de rayonnement et votre corps. Ce transmetteur ne doit pas être placé au même endroit ou utilisé simultanément avec un autre transmetteur ou antenne.

Signed by:

Checked by:

EXHIBIT E

BRING A NEW ERA IN MOBILITY



HL Klemove

HL KLEMOVE SNAPSHOT



2021 Sales:
US\$ 1.1B
KRW 1,216 B



CAGR('17-'21) of total sales:
10.7%



Employees:
1,868+



Countries with
HL Klemove technology
184



49 in **12**
Customers Countries



4
Global manufacturing sites

537 in **9**
R&D engineers R&D sites globally



31
ADAS features in the market

CEO MESSAGE

The recent C.A.S.E (Connectivity, Autonomous, Shared, and Electrification) megatrend has brought new challenges to the automotive industry, calling for transformation.

Therefore, HL Klemove is set to make a big step towards strengthening its positions as an autonomous driving and mobility industry leader. Driven by the solid capabilities of automotive electronics, our state-of-the-art autonomous driving technologies incorporate advanced sensors and S/W algorithms.

Being an industry leader, we are committed to reacting to fast-changing trends with agility to accelerate qualitative growth and consolidate our leadership in the market by regional business expansion.

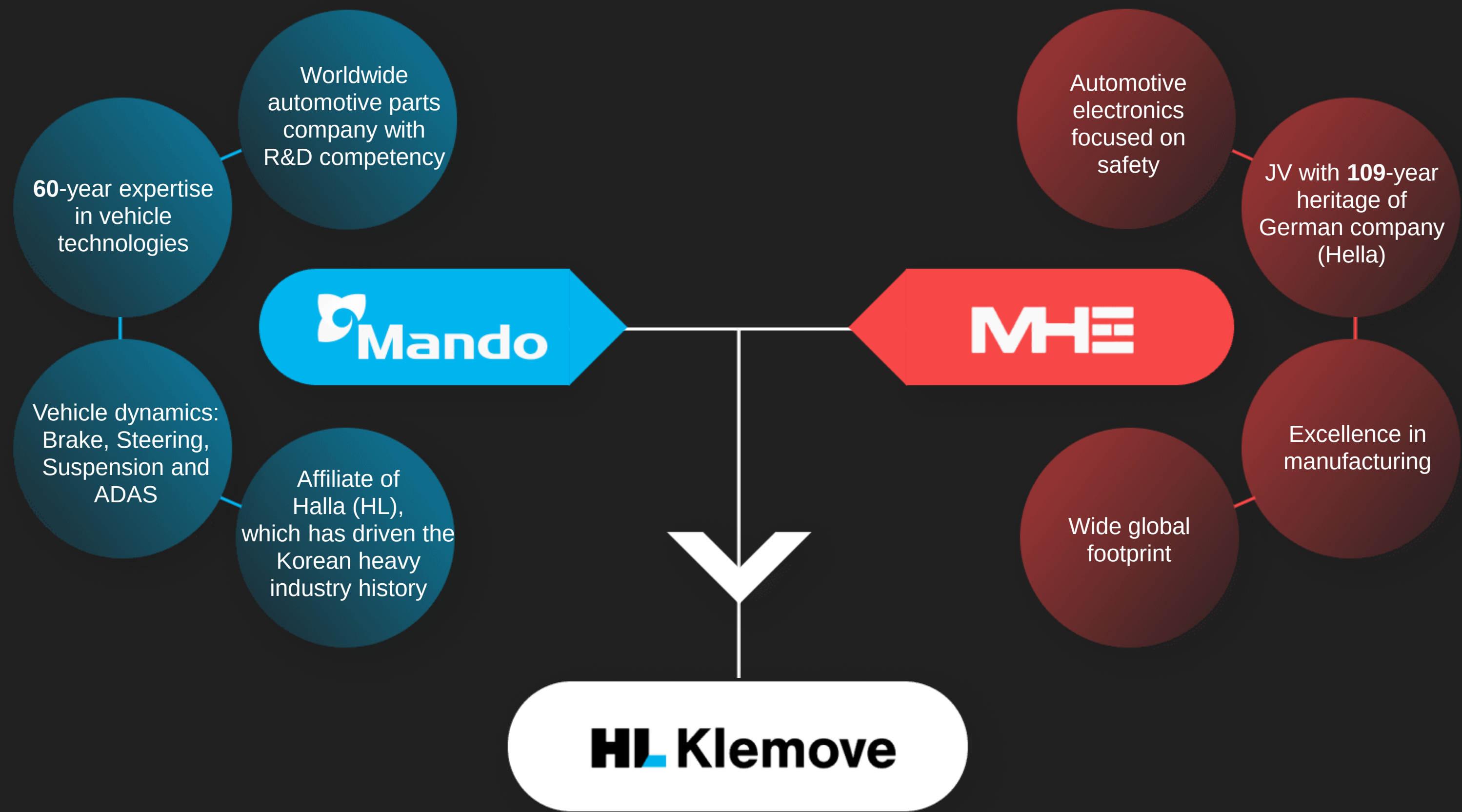
To manifest a world where your most comfortable movement becomes your safest moment, HL Klemove promises a new future based on the fundamentals of having a frontier spirit and “Jeong do” management, moving towards creating more values for our customers and global communities with "Clever Move".

Sincerest Regards,

HL Klemove CEO
Pal-Joo, Yoon



HL KLEMOVE HERITAGE



HL KLEMOVE ESTABLISHMENT



Connectivity



Autonomous

Accelerate the growth of the autonomous driving business through **optimized resource management**



Shared

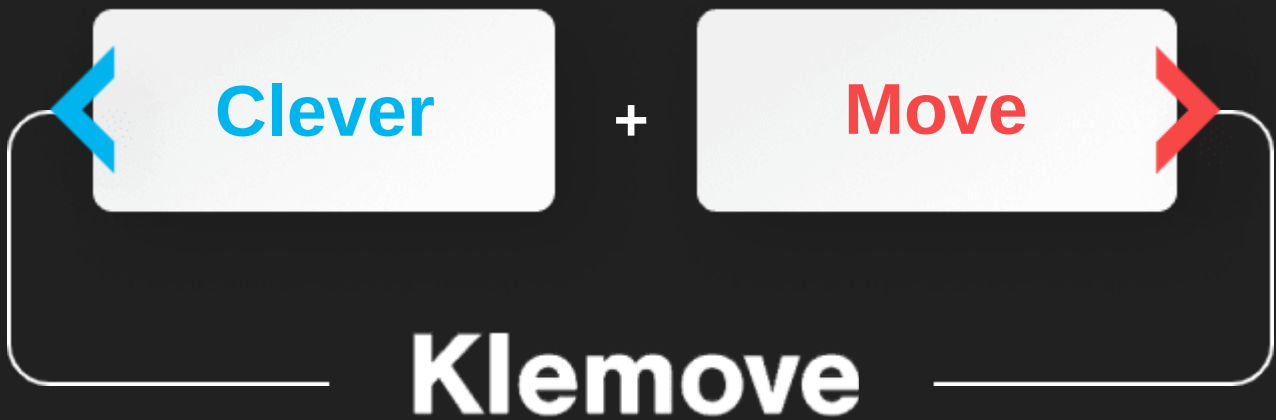


Electrification

Full **value chain internalization** through integration with Mando ADAS and MHE

HL Klemove aims to promote more expertized and optimized business operation in the mobility area by utilizing differentiated business capabilities, strategies, investment priorities, organizations and partnerships

BRAND IDENTITY: FORESIGHT



To move ahead in a clever way, we need to have a “**Foresight**” in technical and management domains.

From an automobile parts company, we will strengthen our position as an innovative pioneer in autonomous driving, AI and fully automated mobility.

HL Klemove

MISSION

To manifest a world where Your most comfortable movement is Your **Safest moment**

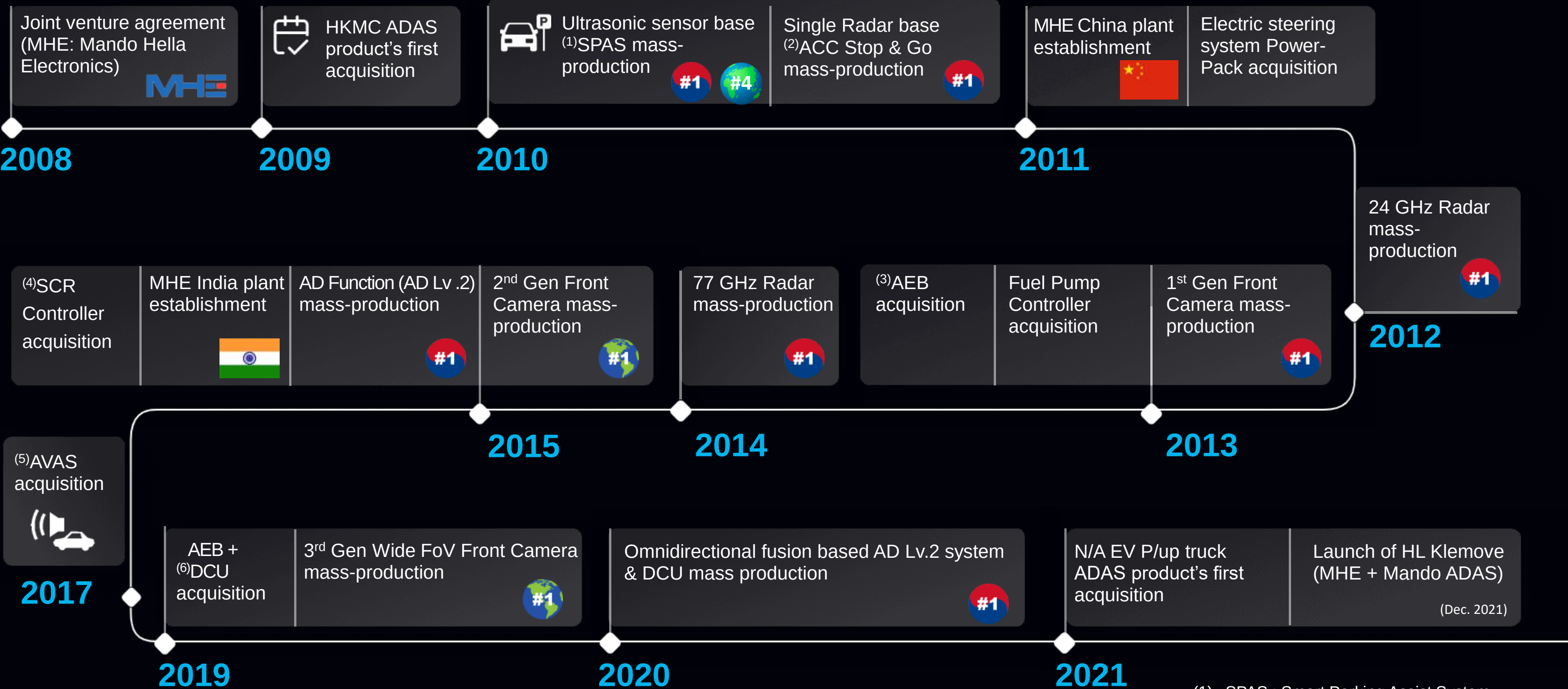
GOAL

The most **Trusted** company in Intelligent Mobility

BRAND IDENTITY

Foresight

HL KLEMOVE FOOTPRINT



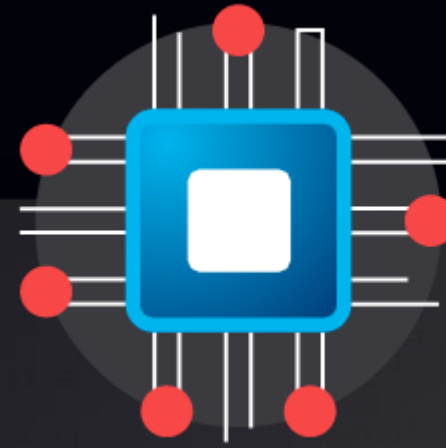
(1) SPAS : Smart Parking Assist System
(2) ACC: Active Cruise Control
(3) AEB: Autonomous Emergency Braking
(4) SCR: Selective Catalytic Reduction
(5) AVAS: Acoustic Vehicle Alert System
(6) DCU: Driving Control Unit

BUSINESS AREAS



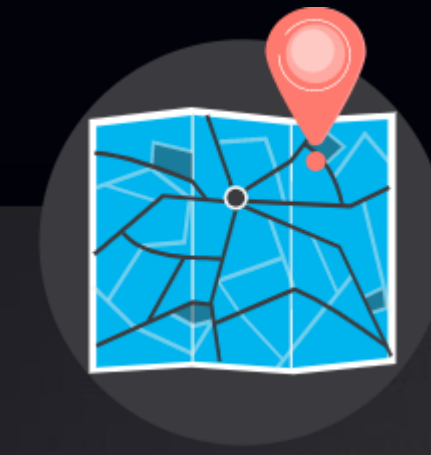
Autonomous Driving

- Autonomous driving solutions
- Active safety solutions
- In-cabin sensing solutions



Automotive Electronics

- Automotive electronics solutions
- ⁽¹⁾AVAS solution for xEV
- Fuel & Urea controller
- ⁽²⁾EMS (focusing on safety-related products)

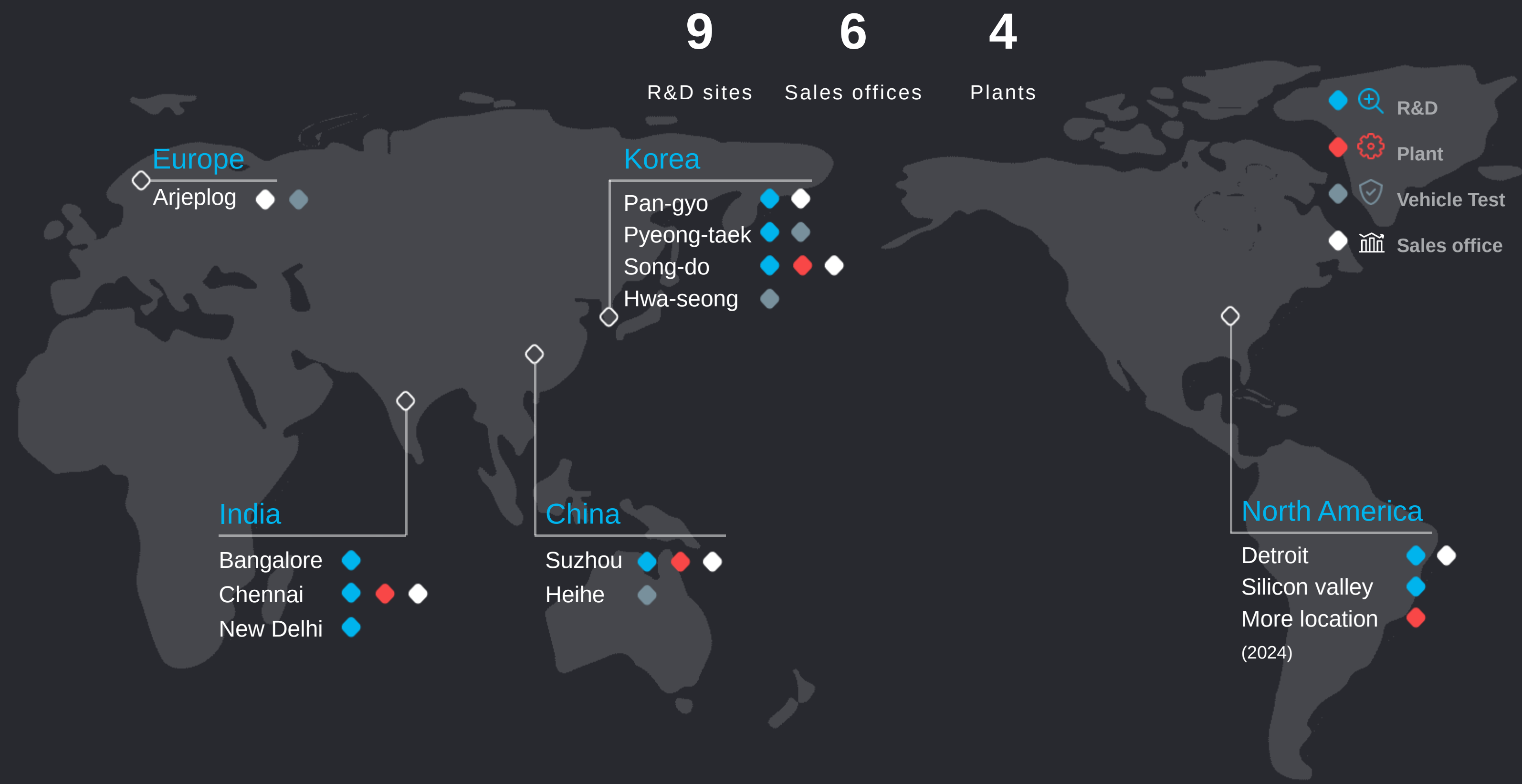


Mobility Platform

- Robot platform
- Data solutions

HL Klemove's autonomous solutions and innovations are aimed to manifest a world where your most comfortable movement is your safest moment

GLOBAL PRESENCE



R&D CAPABILITY & INVESTMENT

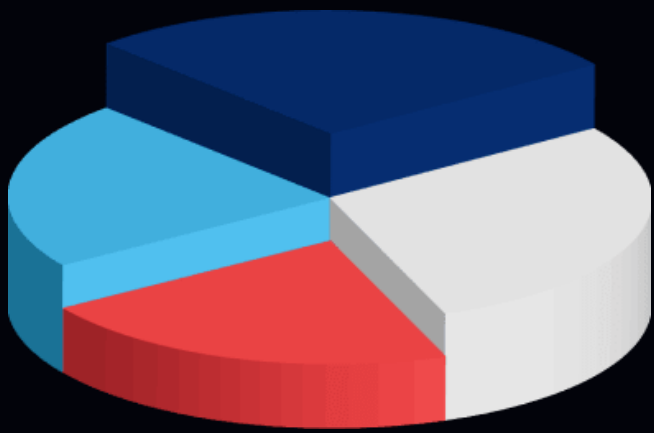
7~8% R&D / Sales

233 Master's & Doctoral degree employees
(21% of domestic workers)

537 R&D engineers (33%) in 9 R&D sites globally

(Dec. 2021)

- HL Klemove holds 2,040 patents and leads the innovation of future mobility



(3Q 2021)

- 587 Domestic patent applications
- 561 Domestic patent registrations
- 453 Abroad patent applications
- 439 Abroad patent registrations

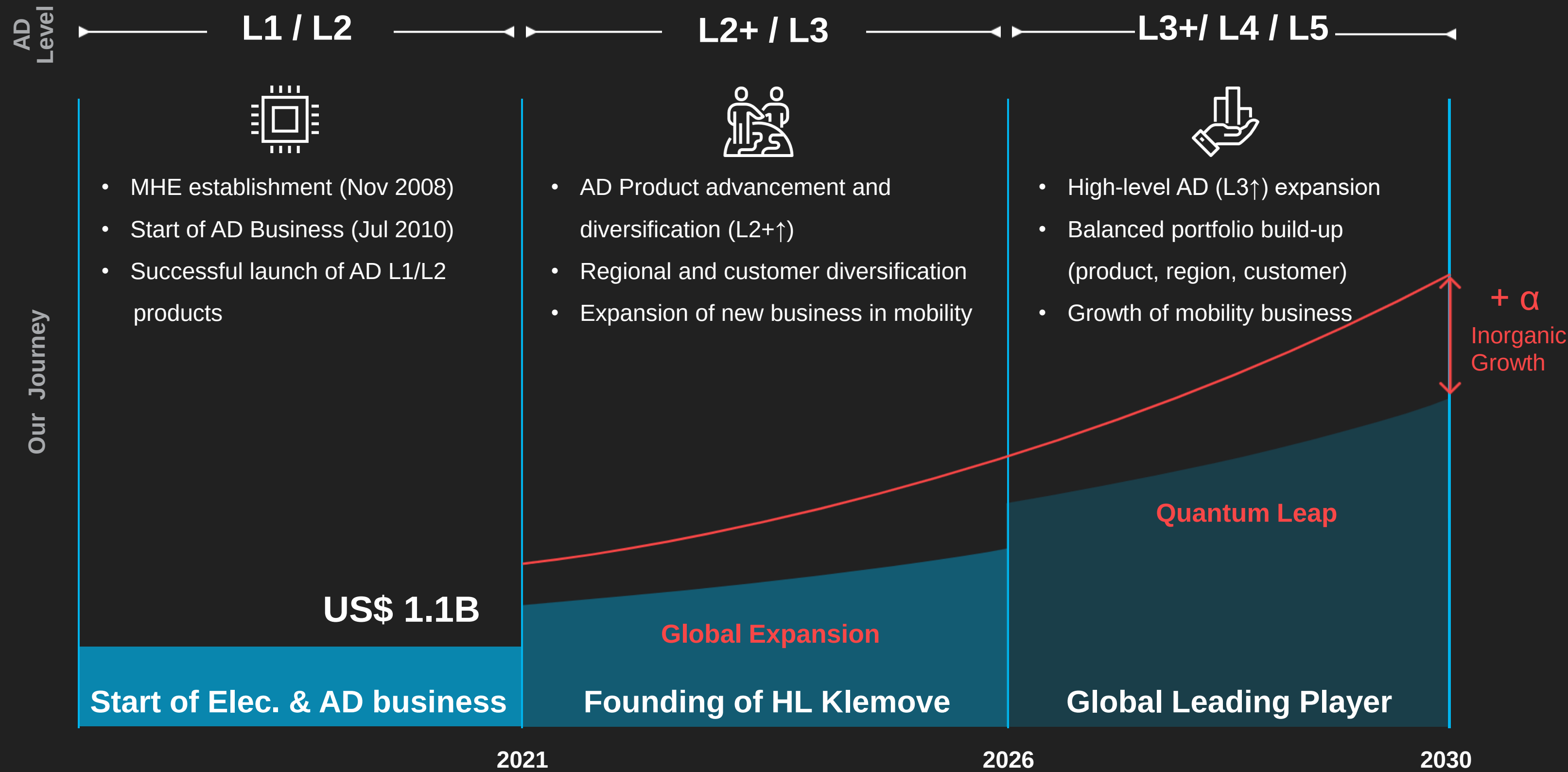


*ICT Patent Management Award 2020

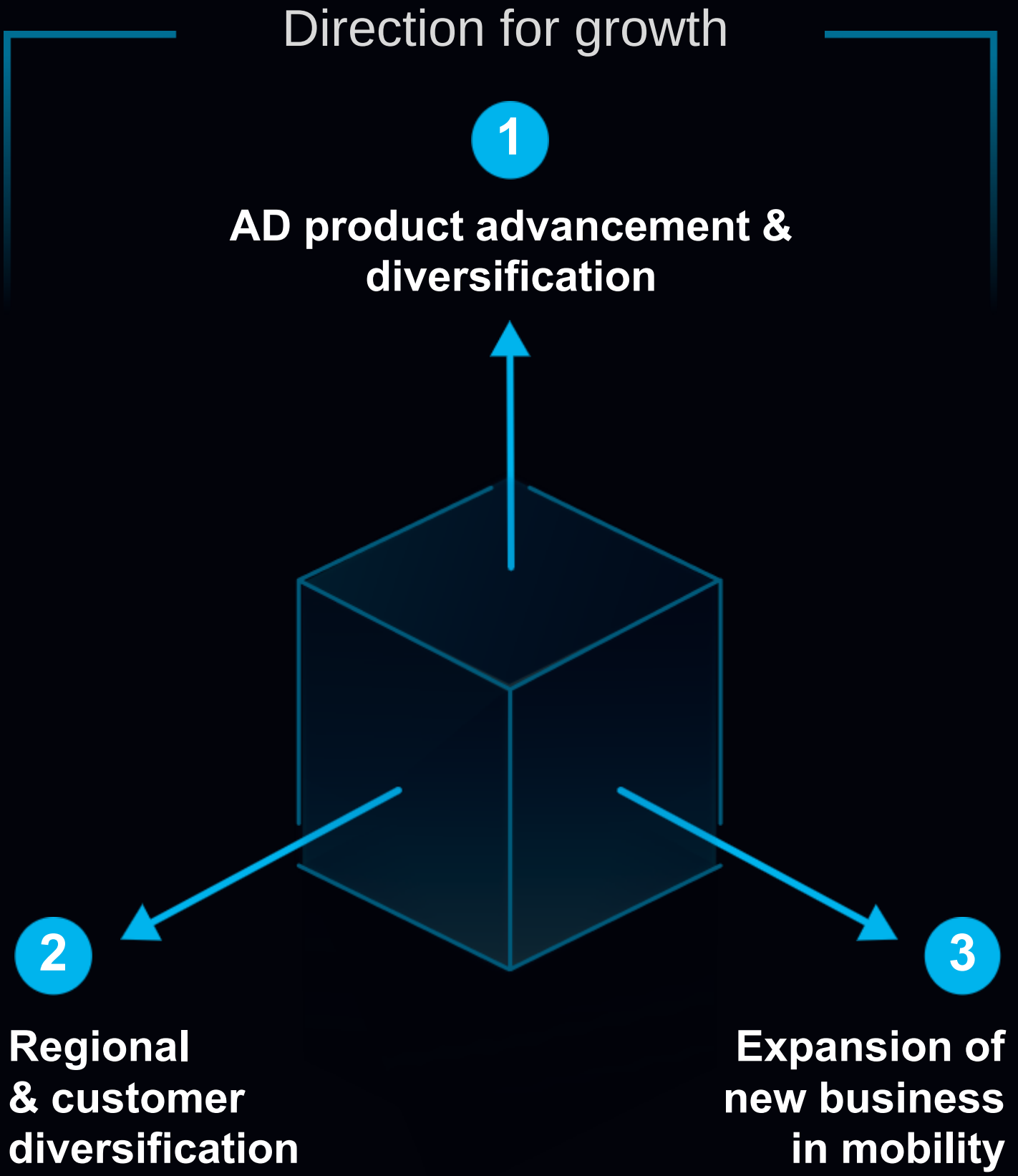


*Information and Communications Technology

JOURNEY FOR SUSTAINABLE GROWTH



GROWTH STRATEGY



- 1 > **Full AD solutions provider**
Responding to advanced AD trend
- 2 > **Innovation with AD technology and SW/data solutions**
- 3 > **Balanced regional and customer portfolio**

WHY CUSTOMERS CHOOSE HL KLEMOVE



01

**GLOBAL MARKET
REFERENCE**



02

**BEST
PERFORMANCE**



03

**EXPERTISE
IN SENSORS**



04

**AGILE
SUPPORT**



05

**MANUFACTURING
EXCELLENCE**



06

**RELIABLE
PARTNERSHIP**

PRODUCT OVERVIEW

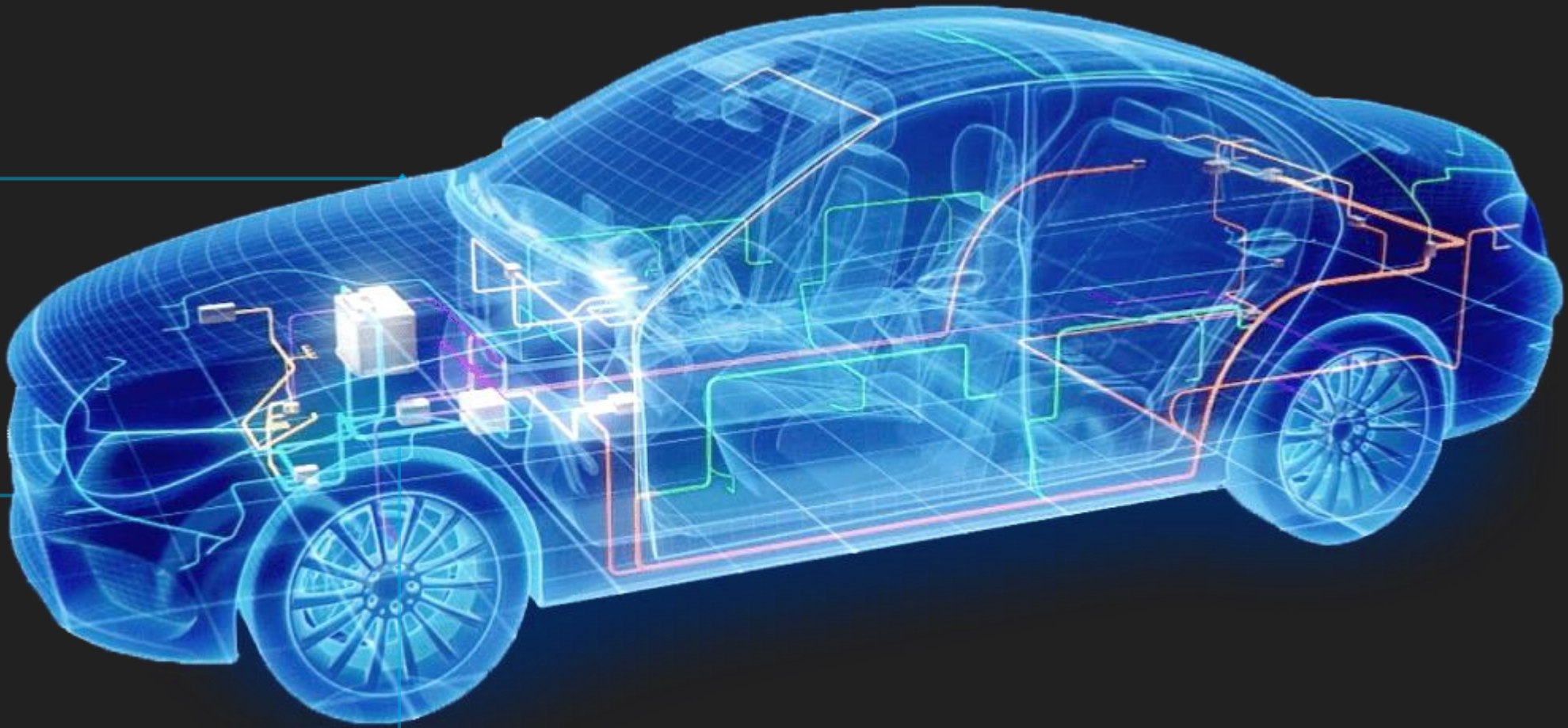


HL KLEMOVE SOLUTION BREAKDOWN

 Sensor = Eye

 Processor = Brain

 Automotive electronics = Nervous system



End-to-End Solutions for future mobility

From the very basic **electrical architecture** for various types of control units to **sensors** detecting both in and outside the mobility ecosystem and to **controllers** to process all the signals, HL Klemove has it all to build highly autonomous, emission-free, and richly featured mobility.

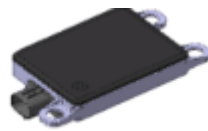
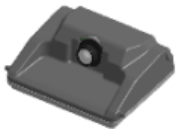
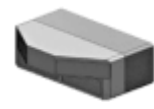
KEY PRODUCTS

1) Autonomous Driving Solutions

System Integration

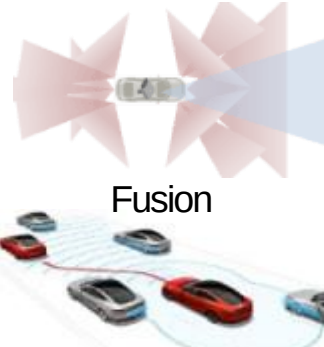
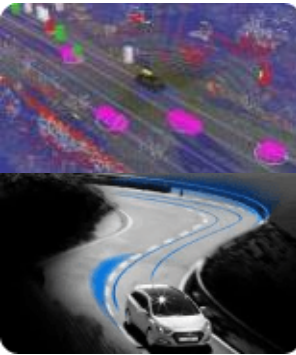
Klemove technology

**PERCEPTION**
: Sensors with ADAS Features

Radar  Front Radar <i>ACC/FCW/AEB</i>	Camera  Front Camera <i>LDW/LKA/HBA/TSR/FCW/AEB</i>	Lidar  Solid state lidar
 Corner Radar <i>BSD/LCA/RCTA</i>	Ultrasonic Sensor  Long Range <i>IPAS</i>  Short Range	

**DECISION**
: DCU with Software

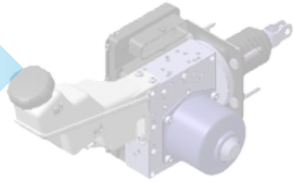
Software

Perception

Fusion
Planning

Control

Hardware

Automated Driving Control Unit (ADCU)

**CONTROL**

ECU → **Brake**


ECU → **Steering**

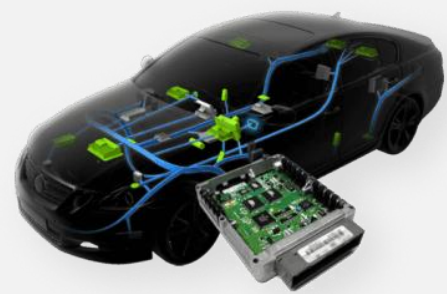

ECU → **Suspension**


KEY PRODUCTS

2) Automotive Electronics

All kinds of automotive electronics solutions,
from design to manufacturing

- 1 **Vehicle dynamics control**
Chassis control unit for braking & steering
- 2 **xEV virtual engine sound system**
Alerting pedestrians to avert accidents
- 3 **Electronics manufacturing service**
Built to print automotive electronics for Tier1/OEM
- 4 **Emission system control**
Zero defects, reliable process



Electronics



Steering ECU

for convenient steering



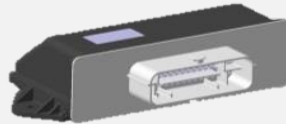
Brake ECU

for safe braking



TOS/TAS

for steering ECU



SCR ECU

for SCR system



FPC ECU

for Fuel pump system



AVAS

*for generating sound
in xEVs*

(1) TOS/TAS : Torque Only Sensor / Torque & Angle Sensor
(2) SCR: Selective Catalyst Reduction
(3) FPC : Fuel Pump Controller
(4) AVAS: Acoustic Vehicle Alert System

LATEST ACHIEVEMENT: GENESIS



LDW, LKA, ELK-Roadedge, HBA, DDA



FCW, AEB –Car/Ped/Cyclist, AEB Plus, AEB-JT/JC

ELK - Oncoming/ Overtaking, AES/ESA



ISA iACC



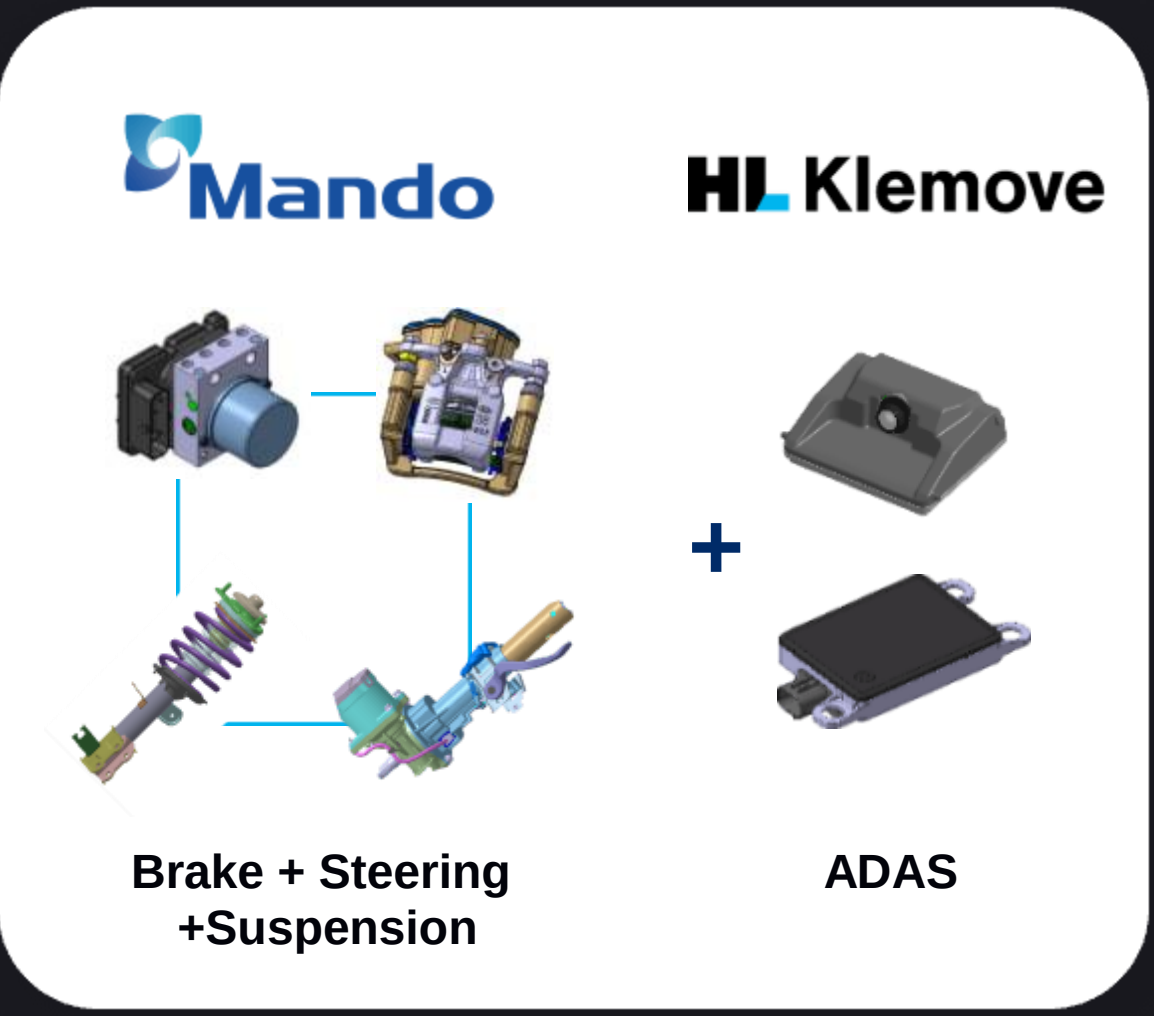
HWA 2.0 (LCS & ALC w/ACC)

NICC, FVSA, ACC Stop & Go,

ACC : Adaptive Cruise Control	AEB : Autonomous Emergency Braking	AES : Autonomous Emergency Steering	ALC : Automatic Lane Change	DDA : Driver Drowsiness Alert	ELK : Emergency Lane Keeping
ESA : Emergency Steering Assist	FCW : Forward Collision Warning	FVSA : Front Vehicle Start Alert	HBA : High Beam Assist	NICC : Navi. Integrated Cruise Control	ISA : Intelligent Speed Assist
LCS : Lane Centering Steering	LDW : Lane Departure Warning	LKA : Lane Keeping Assist	iACC : Intelligent ACC	HWA : Highway Assist	

BUNDLING AND SOURCING BENEFITS

Autonomous Driving Solution
Provider,
Giving **FREEDOM** to OEMs



Development period reduction

- More than 3 months shortening (Ex: EBS-ADAS)
- Project delay risk reduction



Coordination and workload reduction

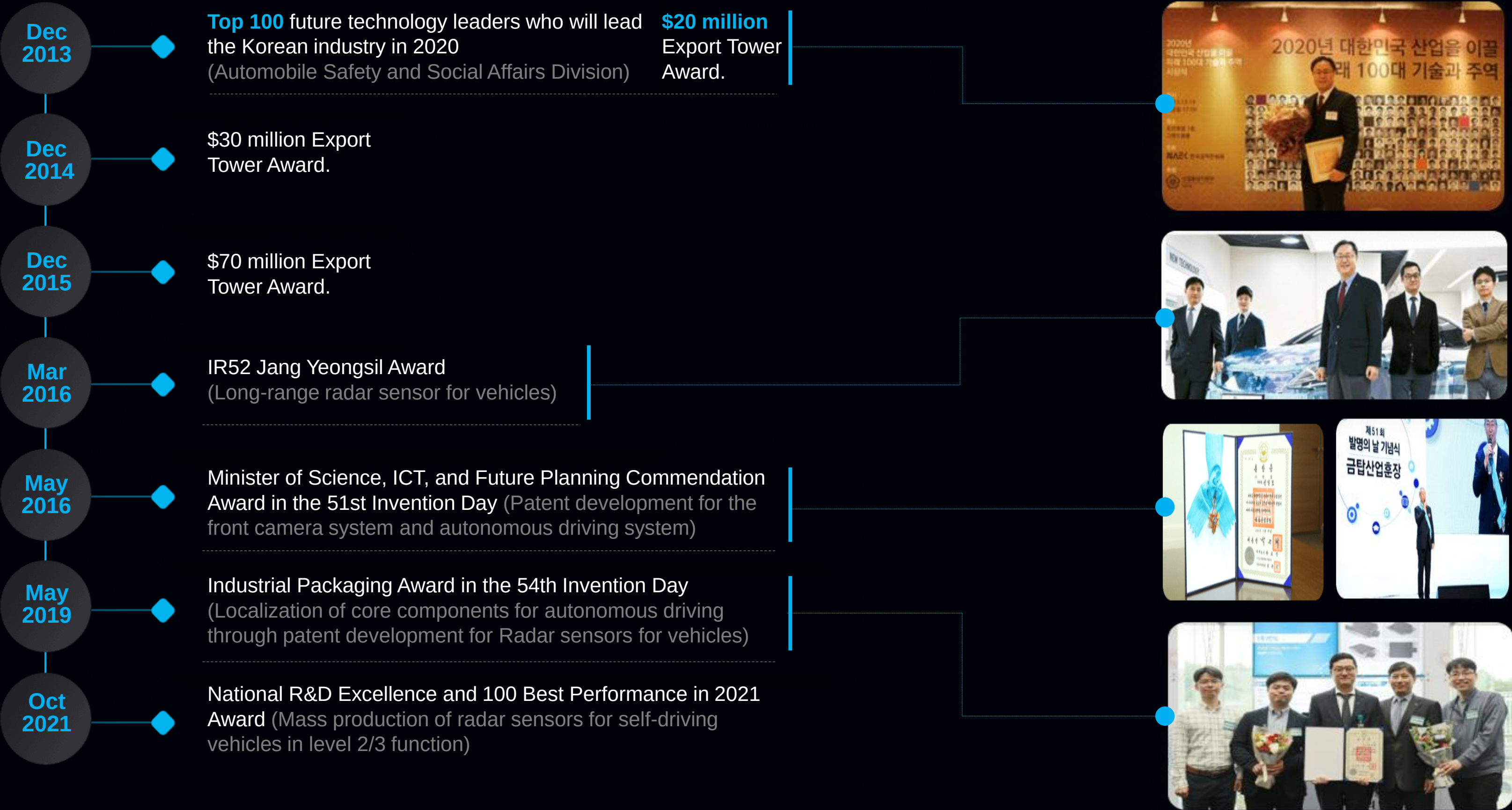
- Interface coordination/ change/ test handling workload reduction
- Responsible / Approval workload reduction



Expertise in vehicle dynamics and AD

- Test vehicle building cost
- Test vehicle shipment cost

AWARDS HISTORY



CUSTOMERS

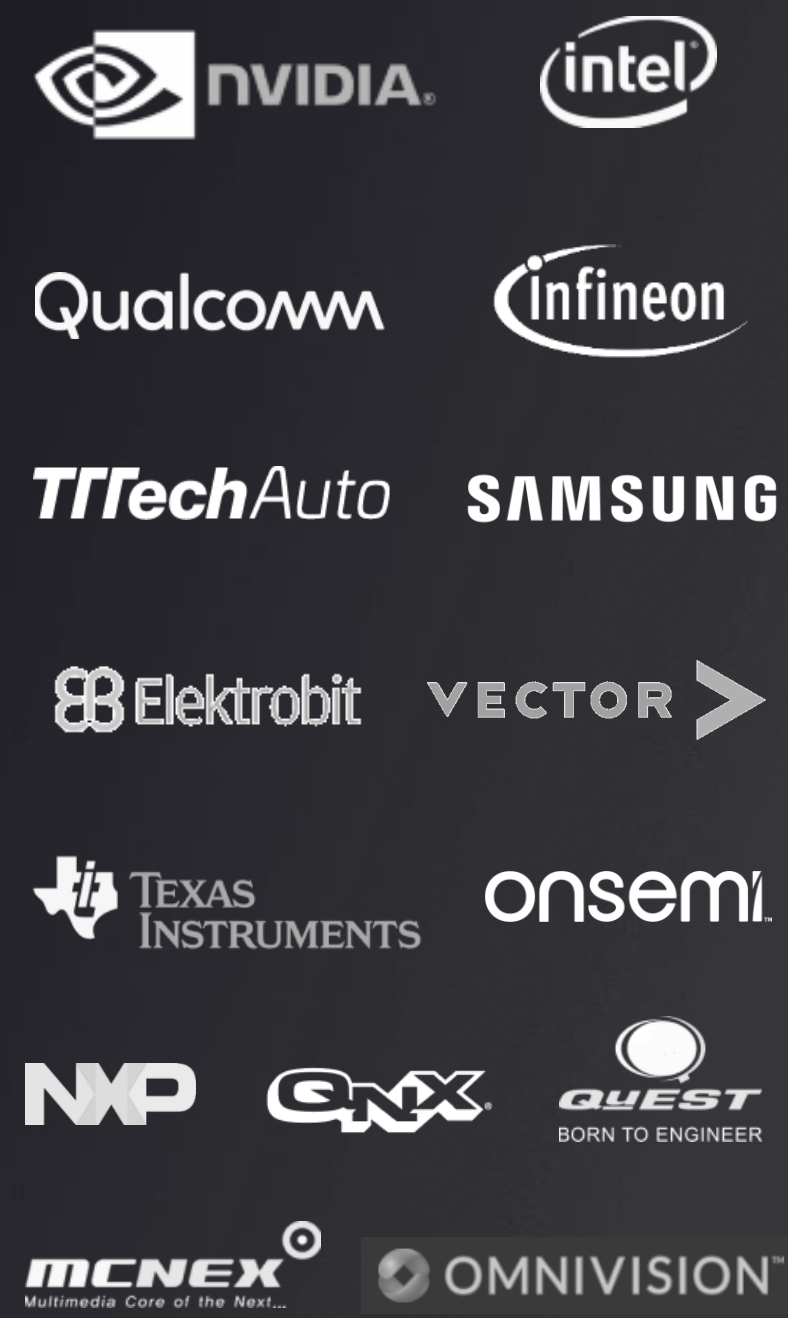


PARTNERSHIPS

Startups

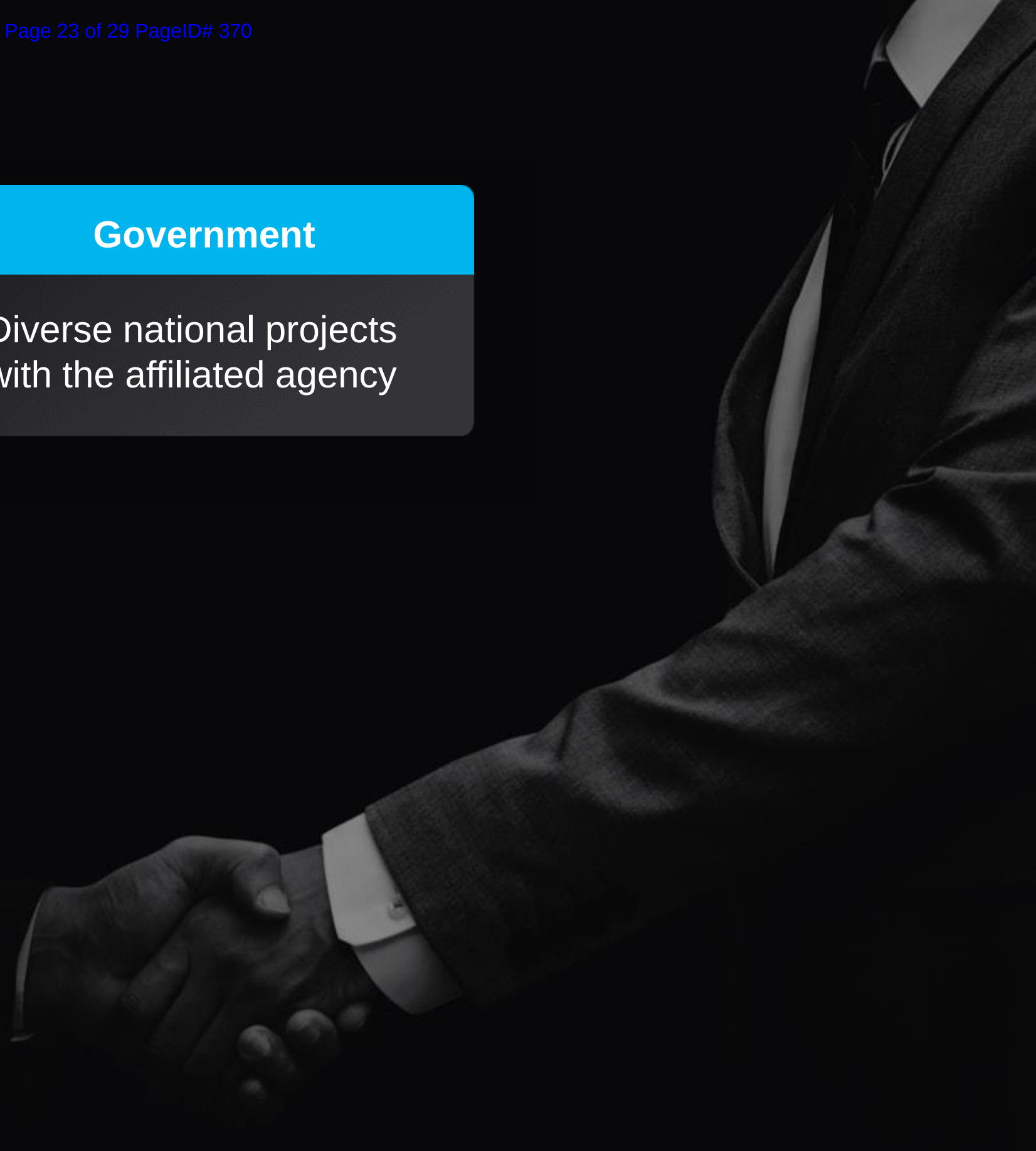


Key industry players



Government

Diverse national projects with the affiliated agency



CERTIFICATION

ISO/TS 16949



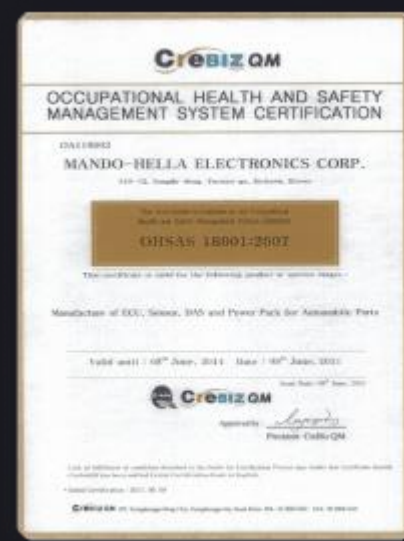
Nov '09 – Jul '18

ISO 14001



Jun '10

OHSAS 18001



Jun '11

ISO 26262



Aug '11

AD permission in KR and CA, USA



ISO 45001



Mar '21

ESD Cert.

ANSI/ESD S20.20 & IEC 61340-5-1



Oct '18

IATF 16949



Jul '18

ASPICE CL2



Dec '13

HL Klemove is continuously

improving all systems and aims to

satisfy customers' expectations

QUALITY & COMPETITIVENESS



Safety Development

- Securing the development and quality based on the APQP process
- S/W verification using 3D virtual reality
- Use of pre-verification to confirm design specifications



Safety Production

- Secure LOT traceability
- Continuous improvement through prediction and prevention
- Secure SMT soldering conditions based on IPC-A-610
- Potential risk reduction through the IQ-FMEA program



Safety System

- Establishing the safety pre-response system
- Quick information sharing and compliance system
- Encouraging enterprise-wide quality management and strengthening capabilities



**“QUALITY” IS THE
CORE VALUE AND GROWTH
ENGINE OF HL Klemove**



SUSTAINABILITY MANAGEMENT



6 MAJOR AREAS OF SUSTAINABILITY MANAGEMENT



01 Technological Innovation



02 Governance / Ethics



03 Human Rights - Safety



04 Environment



05 Supply Management



06 Social Contribution



CONTACT US



www.hklemove.com

Korea (KLK)



224 Harmony-ro,
Yeonsu-gu, Incheon,
Korea



032-850-7114



032-850-7111

China (KLS)



No.160, Kunlunshan
Road, Suzhou New
District, Jiangsu, Suzhou,
China



86-512-8098-7888



86-512-8098-8018

India (KLI)



Plot G 58-60, SIPCOT
Industrial Park, Vallam
Vedagal, Sriperumbudur,
Kancheepuram DT. Tamil
Nadu, India



91-044-6714-1800



91-044-6714-1804

USA (KLA)



29930 Hudson Drive
Novi, MI 48377



1-248-668-4300

HI Klemove