A 20.5 TOPS Multicore SoC With DNN Accelerator and Image Signal Processor for Automotive Applications

Yutaka Yamada, *Member, IEEE*, Toru Sano, Yasuki Tanabe, Yutaro Ishigaki, *Member, IEEE*, Soichiro Hosoda, Fumihiko Hyuga, Akira Moriya, Ryuji Hada, Atsushi Masuda, Masato Uchiyama, Masashi Jobashi, Tomohiro Koizumi, Takanori Tamai, Nobuhiro Sato, Jun Tanabe, Katsuyuki Kimura, Yoshinari Ojima, Ryusuke Murakami, and Takashi Yoshikawa

***Abstract*— Advanced driver-assistance systems (ADASs) that provide machine support to avoid critical accidents are already in wide use in vehicles in today's market. SoCs for such systems have several requirements: 1) high computational performance to run several advanced algorithms at low latency; 2) low power consumption to permit running under the extreme heat and power conditions of real-world vehicles; and 3) high reliability to reduce the risk of serious accidents caused by faults. This article presents a novel SoC for ADAS applications, which resolves these difficult challenges. To achieve high performance with low power consumption, the SoC adopts the heterogeneous architecture, with ten processors, four DSPs, and eight types of accelerators, including the DNN accelerator and the image signal processor (ISP). To achieve higher reliability, the SoC implements several safety mechanisms, including system partitioning to prevent fault propagation, diagnostic features to detect faults, and a dedicated controller to operate runtime built-in self-test (BIST) of the ISP's diagnostic features during the vertical blanking interval (VBLANK). The SoC is implemented in a 16-nm process, and its size is 94.52 mm^2. Peak performance of 20.5 TOPS and low power consumption of 9.78 W are achieved.**

***Index Terms*— Advanced driver assistance system, automated driving system (ADS), deep neural network (DNN), functional safety, image processing, image recognition, image signal processor (ISP), system on chip.**

I. Introduction

ADVANCED driver-assistance systems (ADASs) and automated driving systems (ADSs) are attracting attention for their ability to reduce traffic accidents and make possible driverless vehicles. ADAS features are already in wide use for vehicles in today's market. Traditional ADAS functions, such as autoemergency braking and lane-keeping assistance, are simple vehicle controls of either steering or speed, using braking and acceleration, when risks are detected. Recent ADAS features, such as adaptive cruise control and intelligent parking assistance, operate more advanced vehicle control of both steering and speed. However, these advanced features can only operate in limited areas, such as highways or parking lots. ADAS/ADS systems are expected to become safer and more convenient. For example, future ADAS/ADS systems are expected to support smarter control, such as autoemergency steering and autonomous valet parking, in wider areas, such as city streets and inter-urban areas [1].

ADAS/ADS systems must improve their features in order to make possible safer vehicles. At the research level, various companies and research institutes around the world are developing prototypes and experimenting with the aim of realizing a fully ADS [2]. Technologies for ADAS/ADS systems are actively developed, as the advent of open-source platforms for ADSs [3], [4] demonstrates.

Image recognition and machine learning for object detection and motion prediction are essential technologies for ADAS/ADS systems. It is necessary to improve these technologies, so ADAS/ADS systems with these technologies can detect and track more objects with higher accuracy. Deep learning is well known to achieve higher accuracy than traditional algorithms in a wide area of computer vision [5]–[9]. Deep learning is becoming an important technology for image recognition applications in next-generation ADAS/ADS systems. However, as they improve, image recognition applications for ADAS/ADS systems become more complicated.

As ADAS/ADS systems progress and become able to control vehicles in a broader range of situations, the systems have more responsibility for vehicle control. Moreover, future ADS systems are expected to be even more responsible and make possible fully automated cars [10]. Therefore, it is important to avoid serious accidents even if these systems fail, and in order to do so, it is required to secure the functional safety of vehicles with ADAS/ADS features.

ADAS/ADS applications are, therefore, required for SoCs with high performance for image recognition and providing a high degree of safety. Several SoCs for ADAS/ADS systems have been developed [11]–[16]. In the future, with the progress of ADAS/ADS systems, the requirements on SoCs for higher performance to provide greater safety will further increase.

Manuscript received April 25, 2019; revised August 21, 2019; accepted October 20, 2019. Date of publication November 15, 2019; date of current version December 27, 2019. This article was approved by Associate Editor Hsie-Chia Chang. *(Corresponding author: Yutaka Yamada.)*

The authors are with Toshiba Electronic Devices & Storage Corporation, Kawasaki 212-8520, Japan (e-mail: yutaka6.yamada@toshiba.co.jp).

Color versions of one or more of the figures in this article are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/JSSC.2019.2951391

TABLE I
LEVELS OF DRIVING AUTOMATION

Level		Driving	System request to take over driving	Assistance
0	No Automation	Human	-	Warnings and momentary assistance
1	Driver Assistance	Human	-	Steering or brake/acceleration
2	Partial Automation	Human	-	Steering and brake/acceleration
3	Conditional Automation	System/Human	Yes	Driving the vehicle under limited conditions
4	High Automation	System	No	Driving the vehicle under limited conditions
5	Full Automation	System	No	Driving the vehicle under all conditions

This article presents an SoC for ADAS application. To achieve high performance with low power consumption, the SoC adopts the heterogeneous architecture and consists of ten processors, four DSPs, and eight types of accelerators, including a DNN accelerator and image signal processors (ISPs). Several safety mechanisms (SMs) are introduced to the SoC in order to achieve functional safety. The SoC is implemented in a 16-nm process, and its size is 94.52 mm^2. Peak performance achieved is 20.5 TOPS, and the total power consumption is 9.78 W.

The contents of this article are as follows. Section II introduces the background which includes ADAS/ADS systems and the requirements for ADAS/ADS SoCs. Section III describes the architecture of the SoC and details of its accelerators, and Section IV describes the SMs of the SoC. Section V provides the implementation results and evaluation.

II. SoCs for ADAS and ADS

This section gives an overview of ADAS and ADSs and the requirements for the SoC for these systems.

A. ADAS and ADS

ADASs and ADSs control the movement of vehicles. The systems are similar in that they recognize objects from inputs of several sensors, such as cameras, radars, and LiDARs, and process the inputs to make decisions about the control of their vehicles. However, their scope of responsibility is different.

SAE defined the levels of driving automation [10], as shown in Table I. An ADAS system (level 1 or 2) performs driver assistance only so that a vehicle with an ADAS system requires a driver to control it. An ADS system (above level 3) can drive its vehicle and usually does not require a human to control it. Most vehicles in today's market support only ADAS features. Several companies and institutions are researching and developing ADS features, and it is expected that vehicles with ADS capabilities will be put to practical use in the near future.

Fig. 1 is an example of an ADAS/ADS application. It includes four processes.

1) In the sensing process, the ADAS/ADS system receives information from several sensors.
2) In the perception process, the system detects and classifies objects, such as pedestrians, vehicles, road surfaces, traffic lights, and traffic signs, identifies their positions, and predicts the behavior of surrounding objects.
3) In the planning process, the system determines the path of its vehicle using the results of the perception process.
4) In the control process, the system controls the motion of its vehicle using steering, acceleration, and braking.

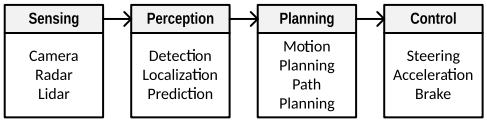


Fig. 1. Example of an ADAS/ADS application.

The sensing and perception processes are important for both ADAS and ADS to detect situations that could lead to a serious hazard. Thus, the ADAS system requires the same or similar performance as the ADS system for these processes, which includes a lot of signal processing and image recognition functions. Both ADAS and ADS require high computing power to perform these functions. On the other hand, as the planning and control processes manage vehicle control, the scope of these processes is different between ADAS and ADS. The difference between ADAS and ADS is mainly the responsibility for vehicle control of the system. The ADAS system assists the driver with notification and may simply control its vehicle, and the ADS system drives its vehicle autonomously. For example, the ADAS system informs the driver and may control its vehicle motion, such as emergency braking if the system detects a situation that could lead to a serious accident. The ADS system also needs to control its vehicle with steering, acceleration, and braking to avoid an accident while continuing to drive if possible in the same situation. Thus, the ADS system performs more complicated calculations in the planning and control processes than the ADAS system does.

As ADAS/ADS systems advance, SoCs for ADAS/ADS systems are also required to have higher performance, and their importance is also increasing. The requirements of these SoCs, especially for performing sensing and perception processes, are: 1) high performance to detect and classify a lot of objects with high accuracy; 2) low power consumption to keep operation stable under severe temperature conditions in vehicles; and 3) functional safety to avoid serious accidents even if some components fail. These are essential in a high-performance

ADAS/ADS system that supports the safe functioning of vehicles.

B. High Performance With Low Power Consumption

As ADAS/ADS systems are expected to support vehicle safety better and become more convenient, systems that meet those requirements will be able to perform more complicated vehicle control. Safer ADAS/ADS systems must be able to recognize and track more objects with more accurate and support more advanced sensors, such as cameras with higher resolution and higher dynamic range. Thus, the SoCs must have the higher computing power for image recognition to perform more advanced algorithms and support such sensors.

The requirement for power consumption of automotive SoCs is also severe because these SoCs are placed in a vehicle environment, sometimes at high temperature. Moreover, even if at normal temperature, it can be difficult to attach cooling fans to SoCs depending on their location, such as behind a front mirror. In this case, it makes the low power requirement even more severe. Therefore, the SoCs for ADAS/ADS systems are required to have high performance with low power consumption.

C. Functional Safety

ADAS/ADS systems grow more advanced and will be able to perform more critical vehicle control. Thus, the reliability of ADAS/ADS systems becomes more important to avoid faults that could cause serious accidents. Furthermore, the SoCs for ADAS/ADS systems are also required to realize functional safety.

The approach to functional safety is to reduce risk, which includes severity and frequency, to an acceptable level. Thus, it is important to reduce the failure rate and/or failure severity at the time of failure occurrence. ISO26262 [17] is the international functional safety standard for electrical and/or electronic systems for road vehicles. The targets of ISO26262 include both overall ADAS/ADS systems and their semiconductors, such as SoCs. ISO26262 describes a safety risk level called the automotive safety integrity level (ASIL), which is classified from ASIL-A to ASIL-D based on the risk assessment. It defines the target failure metrics for each ASIL, as shown in Table II.

The approaches to reduce risk are to reduce failure rates and/or to increase fault detection rates, and introducing SMs is required. SMs include diagnostic features to reduce failure rates and increase fault detection rates. They handle errors and send alerts when a fault occurs. The SoCs for ADAS/ADS systems need to introduce SMs in order to support functional safety.

III. Architecture of the SoC

We have developed image recognition SoCs for ADAS [11], [12]. This article describes how we developed a next-generation SoC for ADAS applications. Relatedly, we also developed the performance estimation environment of this SoC in [18]. This section gives an overview of the SoC and the structure of its image processing accelerator, especially the DNN accelerator and the ISP.

TABLE II
Target Failure Metrics of ISO26262

	ASIL-A	ASIL-B	ASIL-C	ASIL-D
Failure Rate	<1000FIT	<100FIT	<100FIT	<10FIT
SPFM	-	>90%	>97%	>99%
LFM	-	>60%	>80%	>90%

FIT: Failure In Time
SPFM: Single Point Failure Metrics
LFM: Latent Failure Metrics

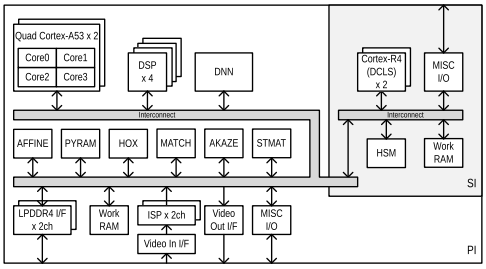


Fig. 2. Overview of the SoC.

A. Overview of the SoC

The target of the SoC is mainly the perception processes of the ADAS application, as shown in Fig. 1. An example of the SoC use case is simultaneous detection of multiple objects using one image sensor of more than 4 Mp or two image sensors of more than 2 Mp. Objects that must be detected in this example are pedestrians, motor vehicles, bicycles, traffic signs and lights, road surfaces, and so on. In this case, the location of the SoC is behind a front mirror, and it is difficult for the SoC to attach a cooling fan.

The overview of the SoC is shown in Fig. 2. As the SoC performs both image recognition processes that need high computing power and control processes that need high reliability, we partitioned the SoC into two regions. One region is called the "processing island" (PI). PI performs image recognition applications and needs high performance with low power consumption. The other region is called the "safe island" (SI). SI performs control applications, such as vehicle control, fault diagnosis and error handling, and needs high robustness.

PI includes two clusters of quad ARM Cortex-A53 application processors, four DSPs for various image processing tasks, eight types of hardware-dedicated accelerators, a two-channel 32-b LPDDR4 interface, 16-Mbits on-chip SRAM, and various I/O interfaces.

SI includes two dual-core lock-step (DCLS) ARM Cortex-R4 control processors, a hardware secure module (HSM) for managing security data, such as cryptographic keys and performing encryption and decryption, 8-Mbits on-chip SRAM, and various I/O interfaces.

In addition, there are several memory protection units (MPUs) to prevent unexpected memory access.

B. Accelerators of the SoC

The SoC has eight types of image processing accelerators to achieve both high performance and low power consumption.

The design approaches are to introduce parallel architecture to reduce the frequency and local memory to reduce DRAM accesses.

The accelerators are as follows: 1) the deep neural network (DNN) accelerator for classification and segmentation; 2) AKAZE accelerator for feature point tracking; 3) STMAT accelerator for matching stereo images; 4) two channels of the ISPs for making color images from raw images; 5) HOX accelerator for detecting various objects using CoHOG [19]; 6) AFFINE accelerator for making affine transformation images; 7) PYRAM accelerator for making pyramid images; and 8) MATCH accelerator for object tracking.

To enhance safety and convenience, the system must detect and track more objects with more accuracy and support more advanced image sensors with higher resolution and higher dynamic range. The DNN, AKAZE, and STMAT accelerators are implemented to perform advanced image recognition algorithms at high speed. The ISPs support the latest sensors that output high-resolution and high-dynamic-range (HDR) images.

The HOX accelerator detects objects by using CoHOG, which is the traditional feature descriptor, and a linear support vector machine (SVM) classifier. Thus, the computation requirements for object detection using CoHOG and SVM are much less than for common DNN. As the HOX accelerator performs the traditional feature-based object detection, the accuracy of this accelerator is sometimes worse than that of the DNN accelerator. However, the throughput of the HOX accelerator is better than that of the DNN accelerator, so the HOX accelerator is suitable for high-volume image detection applications, such as screening. We implemented both HOX and DNN accelerators in the SoC. Note that the HOX, AFFINE, PYRAM, and MATCH accelerators are almost the same as those in our previous work [12].

C. DNN Accelerator

Recently, several image recognition algorithms using DNN have been able to achieve high accuracy in computer vision [5]–[9]. A DNN application called VGG-16 [6] is shown in Fig. 3. The application includes several network layers; each layer has a lot of multiply-accumulate (MAC) calculations. Furthermore, the amount of DNN application data, input features, weight parameters, and intermediate data between layers is huge, so the requirement for DRAM bandwidth is very high. For example, VGG-16 requires over 15.5G MAC operations and 138M weight parameters per inference, which cause massive memory-bandwidth requirements. Thus, we developed the DNN accelerator to execute DNN applications efficiency.

Fig. 4 shows an overview of the DNN accelerator. It consists of a DMA unit for transferring feature volumes and weight parameters from and to DRAM, local memory for storing and loading data and intermediate data and sharing it between network layer operations, a copy/fill unit for initializing local memory and transferring data to and from inner local memory, an execution unit for performing network layer operations, a control unit for overall control of this accelerator, and a configuration buffer for buffering configuration parameters.

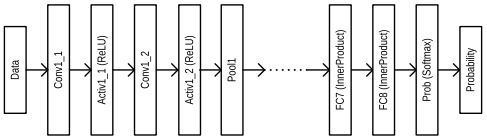


Fig. 3. Example of a DNN application.

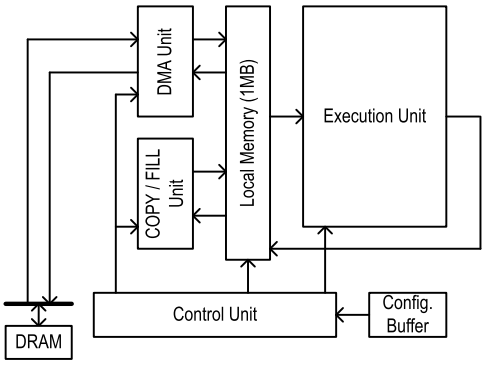


Fig. 4. Overview of the DNN accelerator.

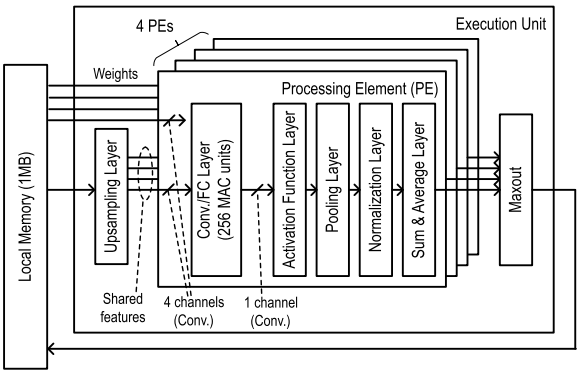


Fig. 5. Structure of the execution unit. The execution unit has four PEs to perform DNN operations. Each PE corresponds to an output channel, so four output channels are processed simultaneously. Dedicated modules for general DNN layer operations are implemented in each PE. In order to reduce data access power consumption by SRAM access, intermediate feature data between the layers implemented as modules are directly forwarded to cascaded modules. For convolution layer calculations, four-channel input feature data are shared among the PEs.

The DNN accelerator uses the following approaches to reduce power consumption: 1) introducing parallel architecture and reducing local memory accesses in the execution unit and 2) reducing DRAM accesses. Here are the details of these approaches.

1) Execution Unit: The structure of the execution unit is shown in Fig. 5. It has four "processing elements" (PEs) that perform simultaneous DNN operations for four output channels. Four PEs can share input features loaded from local memory to reduce local memory accesses. In particular, four input feature channels loaded by the local memory are shared among four PEs when this unit executes convolution layers. Therefore, the four loaded input channels are shared among four output-channel calculations. Each PE includes the following modules for general DNN layer operations: upsampling layer module, convolution/fully connected (Conv./FC)

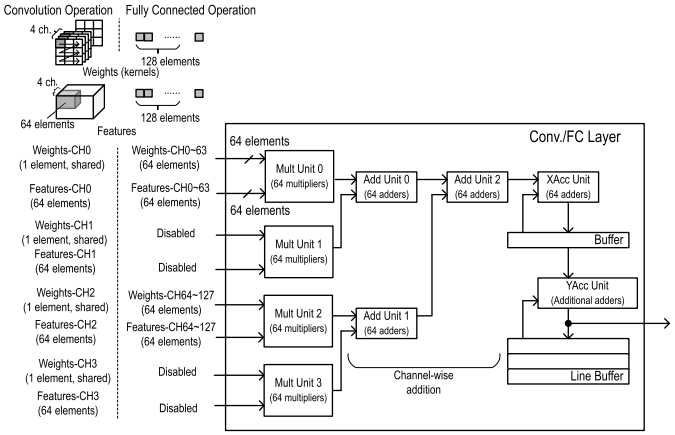


Fig. 6. Structure of Conv./FC layer module in each PE. Each Conv./FC layer module consists of pipelined multipliers, adders, and buffers for accumulation. The Mult and Add units are used for channelwise MAC operations. The XAcc and YAcc units are used to accumulate results of the Add units for corresponding kernel elements for convolution.

layer module, activation function layer module, pooling layer module, normalization layer module, sum and average layer module, and maxout layer module. As the amount of the output data from these modules is huge, it is difficult to store all intermediate data in local memory. These modules are pipelined directly using dedicated buffers and can perform their operations efficiently without storing intermediate data. If running networks contain layers that the accelerator does not support, they are realized by cooperation with DSPs. This method makes it possible to support the latest networks.

The Conv./FC layer module can calculate the convolution layer and fully connected layer operations. The module has 256 MAC units per PE, and four PEs have a total of 1024 MAC units. Its performance achieves 1.6 TFLOPS for the DNN application. The MAC units support the IEEE754 binary-16 format (FP16). Recently, a lot of integer-based DNN accelerators have been proposed, such as in [20] and [21]. Generally, the power efficiency of integer units is higher than that of floating-point units. Although they also could be used for some automotive applications, we chose this format to ensure better accuracy for such applications. The structure of the Conv./FC layer module is shown in Fig. 6.

Convolution operations of the Conv./FC layer module are as follows: The module generates an output feature map, taking the input feature map and the kernel as input, when executing a convolution operation. During initialization, kernel parameters are stored in local memory. Loading of input features and convolution operations can be performed simultaneously to process large input features efficiently. As shown in Fig. 6, by using four sets of 64 adders and 64 multipliers, matrix multiplication using 64 feature elements from four input channels and four weights is calculated for an output channel. A column of the feature matrix corresponds to each input channel. Input features are loaded in the channel direction before loading in the vertical direction so that channelwise accumulations are realized using the line buffers. By applying this method, an input feature that has a large number of channels can be processed efficiently. In particular, pipelines behave as follows.

1) Each Mult unit multiplies input features and kernel parameters of each input channel.
2) Add units 0, 1, and 2 perform convolutional addition of the results of the four Mult units, and the results of 64 elements are generated. The results are convolution values of four input channels.
3) The XAcc unit accumulates the results of Add units over multiple cycles and generates the convolution results along the horizontal direction of the kernel.
4) The YAcc unit accumulates the results of the XAcc unit over multiple cycles and generates the convolution results along the vertical direction of the kernel. It also performs convolutional additions in the direction of the input channels if input features are more than four channels.

For throughput of the pipeline, we implemented two types of accumulation units: XAcc and YAcc.

This layer module supports all combinations of the kernel sizes between 1×1 and 5×5, and the pipeline is designed to achieve the highest MAC utilization when the kernel size is larger than 3×3. Kernels larger than 5×5 can be realized by splitting the kernel and using the execution unit for each split kernel. Input feature map padding is realized by hardware for kernels up to 5×5. Although padding for kernels larger than 5×5, which would be executed as split kernels, is not supported by the hardware, the Copy/Fill operation can be used to generate padded feature maps in local memory. This is a typical use case of the Copy/Fill units. The Copy/Fill unit is also implemented to provide flexibility to accommodate new layers that can be realized by data reallocation in local memory.

The fully connected operation behavior of the Conv./FC layer module is as follows. This module generates an output feature map, taking the input feature map and the weight parameters as input, when executing a fully connected operation. During initialization, input feature volumes are stored in local memory. Weight parameters cannot be stored in local memory because the number of weight parameters for fully connected operation is too large. Loading weight parameters from DRAM and fully connected operations can generally be performed simultaneously. Although the execution unit has 1024 MAC units, it is not efficient for all MAC units to perform fully connected operations. This is because the throughput of loading weight parameters is smaller than the throughput of performing MAC operations by all units. Therefore, only one PE is enabled, and the 128 MAC units of the single PE perform when a fully connected operation is running, as shown in Fig. 6. In this case, the performance achieves 0.2 TFLOPS. The MAC operations calculate 128 elements of input features and 128 elements of weight parameters for each output channel. Weight parameters are updated by loading from DRAM.

1) Each Mult unit multiplies input features and corresponding weight parameters.
2) Data from the Mult units pass through Add units 0 and 1, and the results are added together by Add unit 2. It generates 64 intermediate data elements.

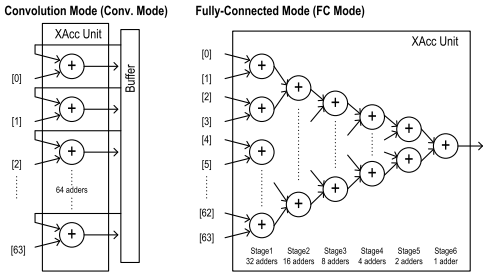


Fig. 7. Two dynamic configurations of the XAcc unit. The XAcc unit consists of 64 adders. It can be configured dynamically as two modes: convolution mode and fully connected mode. In convolution mode, this unit accumulates 64 independent elements for horizontal convolution. In fully connected mode, this unit constructs an adder tree to sum up all elements of input vectors. The adders are shared between the two modes.

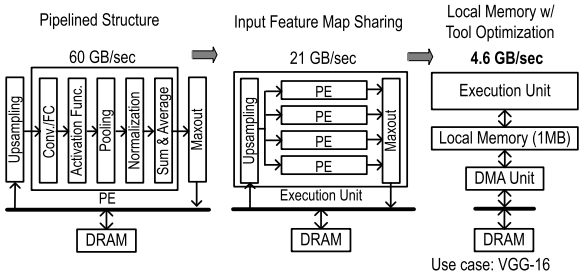


Fig. 8. Example of DRAM bandwidth reduction by using the local memory, network-partitioning tool, and input feature map sharing.

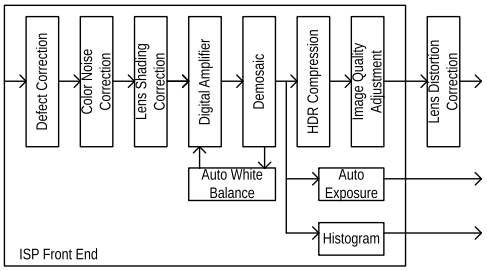


Fig. 9. Overview of the ISP.

3) The XAcc unit sums up the 64 intermediate data elements from Add unit 2 by composing an adder tree, as shown on the right-hand side of Fig. 7.
4) The YAcc unit accumulates the results of the XAcc unit if the input features are greater than 128 elements.

The XAcc unit can support both convolution operations and fully connected operations because the units change internal connections depending on the operation mode, as shown in Fig. 7. Therefore, the Conv./FC layer module can perform both convolution and fully connected operations.

2) Features for DRAM Bandwidth Reduction: The DNN accelerator has two features for reducing the bandwidth of external memory.

1) The decompression circuit in the DMA unit can reduce the DRAM bandwidth by loading compressed weight parameters and decompressing the data. As this circuit uses simple decompression methods such as run length and quantization, its area can be reduced.
2) Local memory, such as a scratchpad memory, can reduce the DRAM bandwidth for transferring temporary data between layer modules.

The DMA unit transfers data between the DRAM and local memory, and the execution unit can access only local memory, not DRAM. Furthermore, this structure can simplify the DMA unit and local memory storage. The size of local memory is large enough, at 1 Mbytes, to store the intermediate data of the assumed use cases.

These features combine to reduce DRAM bandwidth. The DNN accelerator executes VGG-16 in 61 ms/inference. If we assume straightforward execution that does not use any local memory, a huge amount of DRAM data access for feature maps, including intermediate layers, is required: about 60-GB/s DRAM bandwidth for the execution time, as shown in Fig. 8. In this assumption, if we apply input feature sharing among four PEs, we can reduce the bandwidth to 21 GB/s. Moreover, using local memory and our network-partitioning tool only requires 4.6-GB/s DRAM bandwidth. In the same way, the accelerator executes ResNet-50 [9] in 55 m/inference, and then, the bandwidth is reduced from 112 GB/s to only 1.56 GB/s. The network-partitioning tool takes trained weight data and network description data based on Caffe [22] as input and generates binaries to run the DNN accelerator. At compile time, the tool partitions a DNN graph into several sub-graphs to efficiently use local memory and reduce DRAM bandwidth. The partitioning is executed on both inter-layer and intra-layer network topologies. These hardware and tool techniques are essential for accelerators that execute bandwidth-consuming DNN processing.

D. ISP

As ADAS/ADS systems must achieve high safety, the systems are required to continually detect more distant objects regardless of driving conditions, day and night. The systems require high-resolution and HDR image sensors and SoCs to support these sensors [23]. Therefore, we implemented the ISP to support 8 Mp and up to 24-bit HDR image sensors.

Fig. 9 shows an overview of the ISP. The ISP includes a front end for performing general ISP operation, and a lens distortion correction (LDC)u for correcting wide-angle lens distortion. The ISP front end consists of a defect correction unit for reducing shot noise and correcting sensor defects, a color noise correction unit for reducing noise, such as color blur, a lens shading correction unit for correcting lens shading, an autowhite balance unit and a digital amplifier for correcting white balance automatically, a demosaic unit for generating color images from RAW images, an HDR compression unit for reducing color depth by tone mapping, an image quality adjustment unit for adjusting contrast and color and edge enhancement, an autoexposure unit for controlling exposure automatically, and a histogram unit for generating luminance and color histograms as statistical information.

The pixel rate of each ISP is 600 Mp/s, and its performance is over 4.4 TOPS. As all these units are directly pipelined, the ISP can perform all functions on input images from an external image sensor without storing temporary data

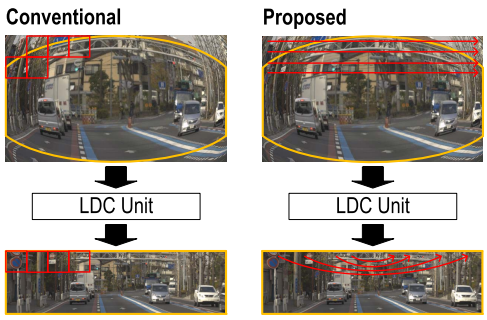


Fig. 10. Image access pattern of the LDC unit.

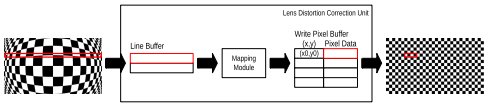


Fig. 11. Structure of the proposed LDC unit.

to DRAM. The ISP achieves low latency and low DRAM bandwidth. Introducing the LDC unit achieves large DRAM bandwidth reduction.

1) DRAM-Less LDC: LDC is to correct the distortion caused by wide-angle lenses, such as fisheye lenses. LDC operations transform input images using preset distortion correction parameters. Wider angle lenses are required for automotive applications because these lenses can detect objects to the left and right of a vehicle better. However, wide-angle lenses are prone to distortion so that LDC operation is needed for automotive applications.

We developed a DRAM-less LDC unit to reduce DRAM bandwidth. Fig. 10 shows the conventional LDC unit and the proposed LDC unit. As the LDC transform images using preset parameters, the region of the input image corresponding to the area of the output image is known in advance. To perform an LDC operation, only the corresponding area of input image needs to be loaded. Therefore, the conventional LDC unit loads only the corresponding area of input images in tiling order, and this method is efficient for LDC operation. However, since image sensors and ISP front ends transfer an image data in raster scan order, DRAM is needed to transfer between ISP front ends and the LDC unit, and large DRAM bandwidth is required to perform LDC operations.

To reduce DRAM bandwidth, we proposed LDC unit support to receive an image in raster scan order. The structure of the proposed LDC unit is shown in Fig. 11. The proposed LDC unit consists of the line buffer for buffering input images, the mapping module for transforming input pixels to output pixels and generating their positions by preset transform parameters, and the write pixel buffer for buffering output pixels and their positions. The write pixel buffer stores the output pixels and outputs them for every fixed number of consecutive pixels. The proposed LDC unit can directly receive the output images from the ISP front end and does not require transfer via DRAM. For 2880 pixels $\times$ 1860 pixels @ 40-frames/s case, this unit can reduce DRAM bandwidth by 2.56 GB/s for one channel and 5.12 GB/s for two channels across the SoC.

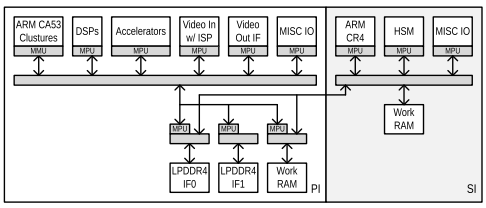


Fig. 12. MPU placement in the SoC.

Both the conventional method and the proposed methods can realize DRAM-less LDC units. The conventional method requires larger buffers than the proposed method. The conventional method requires a large line buffer to store a large amount of input data, but the proposed method does not require a large line buffer because the proposed method converts the input data sequentially and sequentially outputs the data to DRAM. The on-chip SRAM size required to realize that the DRAM-less LDC unit is 6.2 Mbytes for the conventional method and 0.6 Mbytes for the proposed method. Therefore, the proposed method can realize DRAM-less LDC unit efficiently.

IV. SMs of the SoC

SMs are introduced to enhance functional safety and to reduce the risk of serious accidents. When a fault occurs in a component of the SoC, it can lead to a serious accident if it is unhandled. When a fault occurs in a normal component of the SoC, for example, it can propagate a failure of more critical components and lead to more serious accidents. SMs are essential to ensure functional safety. They need to detect faults at the right time, generate alerts, and prevent the propagation of faults to other components.

This section describes the SMs of the SoC, the partitioning feature for prevention of fault propagation, the diagnostic features for fault detection, and the control of runtime built-in self-test (BIST) as an example of diagnostic features.

A. Critical Region Partitioning Feature

As mentioned in Section III, we partitioned the SoC into two regions, PI and SI, to realize both high performance with low power consumption and enhanced safety. As SI performs more important processes than PI, SI must be more reliable. Thus, we designed SI to be more robust than PI and not to propagate faults from PI. For example, the SoC has the mechanisms for monitoring memory access. The bus masters in SI can access the bus slaves in PI; however, the bus masters in PI cannot access the bus slave in SI to avoid unexpected memory access.

Memory management units (MMUs) and MPUs exist in the bus masters and in front of LPDDR interfaces and Work RAM of PI, as shown in Fig. 12. The MMU or MPU in each bus master can prevent propagating undesirable accesses from its master to the interconnect. The MPUs in front of LPDDR interfaces and Work RAM can prevent the masters of PI from accessing SI-allocated regions. The correspondence between the bus masters and the accessible bus slaves is shown in Table III. Therefore, the MMUs and MPUs can avoid unexpected memory access from bus masters.

TABLE III
MEMORY ACCESSIBILITY OF THE SoC

Target	Work RAM In SI	Work RAM In PI	LPDDR4 IF0	LPDDR4 IF1
Master in SI	Not protected	Not protected	Not protected	Never access
Master in PI	Never access	Protected	Protected	Protected

TABLE IV
DIAGNOSTIC FEATURES OF THE SoC

Target	PI	SI
Random Logic	Runtime BIST	Duplicated Logic
Memory	Parity / ECC	ECC
Memory Access	MPU	Duplicated MPU
Clock & Voltage	Monitor	Duplicated Monitor
Bus Access	ECC with Payload	ECC with Payload

B. Diagnostic Features

The SoC has several diagnostic features to detect faults. As SI requires more reliability than PI, we designed the SMs of SI to be more robust than those of PI. The diagnostic features of PI and SI are shown in Table IV.

The runtime logic BIST units are used to realize the diagnostic features for random logic in PI. The duplicated logic and their monitor circuits are used in SI. BIST technology is used for shipping tests and can achieve a high fault coverage ratio; however, runtime BIST has limited time for testing, so it can only perform a limited test pattern and it is difficult for it to achieve high coverage. Therefore, the duplicated circuit can always monitor logic faults and are more robust than the logic BIST unit for runtime diagnostics.

To realize the diagnostic features for memory circuits, parity circuits or ECC circuits are used in PI. As ECC is a stronger error correction method than parity, only ECC circuits are used in SI. The units that generate data used in multiple units, such as the ISP, use ECC circuits to avoid common cause failures.

For diagnosis of memory access, clock, and voltage, MPUs and monitor circuits are used in PI. In SI, MPUs and monitor circuits are duplicated in order to be more reliable.

The ECC code is attached to the payload for diagnosis of interconnect access in both PI and SI because the interconnects of both PI and SI connect a lot of masters and slaves. PI and SI have the same reliable diagnostic features to avoid common cause failures.

C. Control Sequence of Runtime Logic BIST

The logic BIST unit injects the test pattern into flip-flops of the target logic and detects a failure by verifying the result values against the expected values. As the logic BIST unit overwrites all flip-flops, it can execute only while the target logic is not working.

1) Runtime Logic BIST for Accelerators: The processors generally control the behavior of the accelerators, including the DNN accelerator, in the SoC. These processors can control the state of the accelerators including idle state, so the processors also control the runtime logic BIST operation for the accelerators. The control flow of the runtime logic BIST operation for the accelerators is shown in Fig. 13.

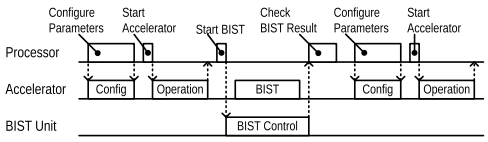


Fig. 13. Timing charts of runtime BIST for the accelerators.

1) The processor configures the parameter of the accelerator.
2) The processor indicates the operation start to the accelerator, and the accelerator starts its operation.
3) The accelerator finishes its operation and sends the finish signal to the processor.
4) The processor indicates the start of the BIST operation to the BIST unit.
5) The BIST unit finishes its BIST operation and sends the finish signal to the processor.
6) The processor confirms the BIST results and performs error handling if a fault is detected.
7) Continue next operation.

The runtime logic BIST operations do not have to be performed after every accelerator operation if it can detect a fault within a fault-tolerant time interval (FTTI). For example, the runtime logic BIST operation may be performed only after the last operation of each frame. As the logic BIST operations overwrite and initialize all flip-flops, all parameters of the accelerator have to be configured after every runtime logic BIST operation.

2) Runtime Logic BIST for the ISP: The ISP starts its processing when it receives frame data from an external image sensor. Then, when the processing of one frame's data is completed, the ISP switches to the idle state. This idle period between two consecutive frames is called the vertical blanking interval (VBLANK). There are several issues related to managing the fault diagnosis of the ISP during VBLANK.

It is difficult for the processors to monitor the operation status of the ISP because the start of the ISP operation is controlled by an external image sensor, not the processors. Therefore, it is also difficult for the processors to control the runtime logic BIST operation for the ISP when the ISP is in the idle state. As a runtime diagnostic feature other than runtime logic BIST circuit, duplication of the ISP logic can be considered. However, the area cost of duplication is so high that this option is also difficult. Thus, the runtime logic BIST is not easy to realize but efficient from the area cost point of view.

The ISP has a lot of parameters for its operations. All parameters have to be reconfigured after runtime logic BIST operations because the BIST operation initializes all flip-flops. It is difficult for the processor to re-configure a large number of parameters between the end of one BIST operation and the start of the next frame data input.

TABLE V
COMPARISON OF RUNTIME DIAGNOSTIC METHODS

	No Diagnostic Feature	DCLS	BIST for Acc. (Sec.IV-C1)	Proposed (Sec.IV-C2)
Fault Detection Coverage	None	High	Middle	Middle
Logic Overhead	1.00	> 2.00	< 1.05	< 1.10
Diagnostic Time	None	None	< 5ms	< 3ms
Necessity for Software Control	No	No	Yes	No

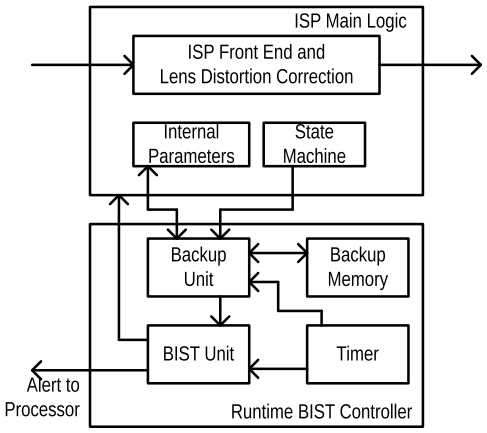


Fig. 14. Structure of runtime BIST controller for ISP.

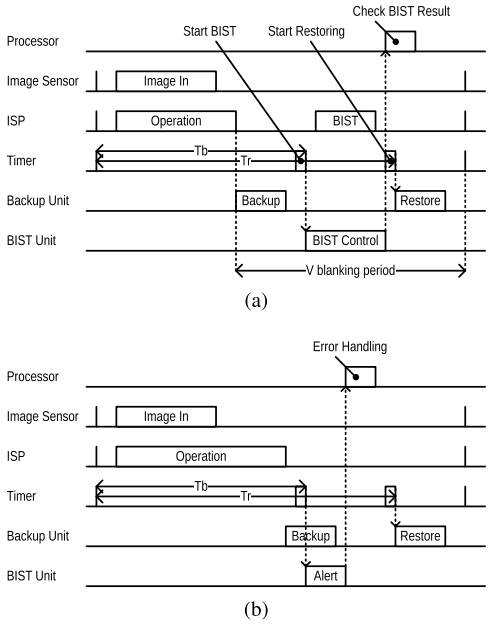


Fig. 15. Timing charts of runtime BIST for the ISP. (a) Runtime BIST can be executed. (b) Runtime BIST cannot be executed.

To solve these problems, the dedicated controller for runtime logic BIST operation of the ISP is introduced. This controller can perform the runtime logic BIST operation during the VBLANK. Furthermore, the controller can back up and restore the parameters of the ISP before and after BIST operations in order to quickly re-configure all parameters after a BIST operation.

The structure of the dedicated controller for the ISP is shown in Fig. 14. This controller includes the backup unit for controlling backing up and restoring, the backup memory for storing all parameters of the ISP, the BIST unit for controlling BIST operation, and the timer unit for notifying start timing of BIST operations and restoring operations. The control flow of the runtime logic BIST operation for the ISP is shown in Fig. 15(a), and the sequence is as follows. In initialization, the processors have to configure all parameters of the ISP before the external image sensor starts processing.

1) The ISP starts its operation when it receives frame data from an external image sensor.
2) The backup unit backs up the parameters of the ISP to the backup memory after the ISP finishes the operation.
3) After the backup unit finishes backing up and the timer unit sends the signal of the BIST operation start-at time "Tb," the BIST unit starts the runtime logic BIST operation.
4) After the BIST operation is finished, the BIST unit sends the finish signal to the processor, and the processor can verify the result of the BIST operation.

TABLE VI
PERFORMANCE OF THE SOC

	Frequency [MHz]	Performance [GOPS]	Power [mW]	Efficiency [GOPS/W]
ARM CA53	1,000	256	1,286	199.1
DSP	500	1,024	1,377	743.6
DNN	780	1,597	1,593	1,002.8
ISP	600	8,880	1,579	5,623.8
STMAT	600	3,480	419	8,305.5
AKAZE	600	2,681	726	3,692.6
MATCH	600	1,175	254	4,627.6
PYRAM	600	396	130	3,046.2
AFFINE	600	204	281	726.0
HOX	600	842	1,696	496.7
ARM CR4	1,000	1	435	2.8
Total		20,537	9,776	2,100.8

Power consumption includes functional safety logics.
Vdd = 0.8V; Process = center; Temperature = 25°C

5) The timer unit sends the signal of the restoring start-at time "Tr," and the backup unit starts to restore the parameters from the backup memory to the ISP.

TABLE VII
COMPARISON OF STATE-OF-THE-ART SYSTEMS

	This work	Our Prev. work ISSCC 2015 [12]	JSSC 2017 [15]	ISSCC 2017 [20]	ISSCC 2018 [21]
Application	Automotive	Automotive	Automotive	Embedded	IoT
Process	16nm	40nm	65nm	28nm	65nm
Peak Performance [GOPS]	20,537	1,900	502	751	364
Power Efficiency [GOPS/W]	2,101	564	862	2,930	3,080
Area Efficiency $\left[\mathrm{GOPS/mm^2}\right]$	217.3	18.0	31.4	21.5	21.6
Normalized Area Efficiency	217.3	65.9	239.3	52.2	347.6

Normalized area ratio (16nm : 28nm : 40nm : 65nm) assumed to be (1 : 2.43 : 3.66 : 11.08).

6) After restoring is finished, the ISP is ready to perform its operation with the data from the next frame.

"Tb" and "Tr" are the parameters that the dedicated controller uses to notify the start times of BIST operation and restoring. The meaning of these parameters is as follows.

1) The parameter "Tb" indicates the period from the start time of frame operation to the expected start time of BIST operation.
2) The parameter "Tr" indicates the period from the start time of frame operation to the expected start time of the restore operation.

The ISP operation, as the main function, takes precedence over the BIST operation as the diagnostic function. The BIST operation will not be performed if the BIST operation cannot be completed before the ISP operation for the next frame is started. For example, in Fig. 15(b), completion of the ISP operation is delayed due to bus delays and so on. In this case, the BIST operation is not performed, and the error signal to indicate that the BIST operation was not performed is sent to the processor.

The dedicated controller detects whether the BIST operation can be performed, using two parameters "Tb" and "Tr" in Fig. 15. The values of "Tb" and "Tr" can be calculated from the following equations:

$$\mathrm{Tb} = P_f - P_b - P_r$$
$$\mathrm{Tr} = P_f - P_r$$

where P_f means the period of frame operation, P_b means the period of BIST operation, and P_r means the period of the restore operation. If P_f, P_b, and P_r are assumed to be 25, 1, and 0.5 ms, then "Tb" and "Tr" are 23.5 and 24.5 ms, respectively.

Introduction of this controller makes it possible to diagnose the ISP using the runtime logic BIST.

Table V shows the comparison of random logic diagnostic methods at runtime. The DCLS method achieves a high fault coverage ratio and does not require diagnostic time overhead. However, its implementation cost is very high because the area overhead is more than doubled. The BIST is a reasonable method when balancing the fault coverage ratio and area overhead though the BIST method is slightly inferior to DCLS in fault coverage. The BIST method mention in Section IV-C1 has a small area overhead but requires software control by a processor. The time overhead due to backing up and restoring is large in this case, so it is difficult to use this method when performing BIST operation for ISP. The proposed BIST method shown in Section IV-C2 is a suitable method for diagnosis of circuits that need to execute BIST processing during a limited time, such as ISP.

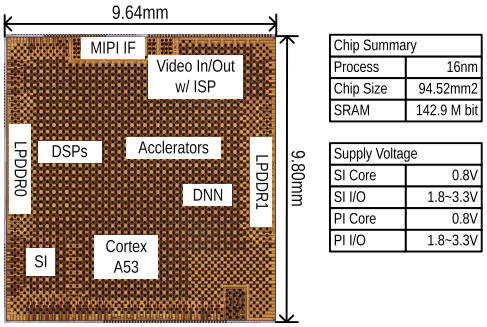


Fig. 16. Chip micrograph of the SoC.

V. IMPLEMENTATION AND EVALUATION

The SoC is implemented in a 16-nm process, and Fig. 16 shows the chip micrograph. Its size is 94.52 $\mathrm{mm^2}$ (9.64 mm × 9.80 mm) and the total on-chip SRAM size is 142.9 Mb. The core voltage is 0.8 V, and the IO voltage is 1.8–3.3 V.

The performance of the SoC is shown in Table VI. Peak performance of the SoC achieved, using ten processors, four DSPs, and eight types of accelerators, is 20.5 TOPS; total power consumption is 9.78 W, and power-performance efficiency is 2.1 TOPS/W. The power consumption given in Table VI was measured for fully utilized condition by simulation. Power consumption of PI measured by the evaluation board is 2.73 W when executing five automotive applications, pedestrian detection, vehicle detection, traffic signal recognition, and lane detection, by two cores of ARM Cortex-A53, four DSPs, and six types of accelerators: the DNN, HOX, AFFINE, PYRAM, MATCH, and AKAZE.

Table VII shows the comparison of state-of-the-art chips [12], [15], [20], [21]. In comparison with SoCs for automotive application, our SoC achieves up to 40× performance and up to 4× power-performance efficiency. In comparison with SoCs for embedded and IoT applications, our SoC achieved

up to 60× performance but delivered about two-third power-performance efficiency. Power consumption is more important for embedded and IoT applications than automotive applications, and the SoC for these applications prioritizes power consumption over performance.

The failure metrics of the SoC were analyzed when the SMs shown in Table IV were applied. The fault coverage ratios of the SMs are estimated with reference to the information in the ISO 26262 standard. For example, the memory fault detection ratio by ECC and the random logic fault detection ratio by DCLS are assumed to be 99% and 99% or more, respectively. Also, the random logic fault detection ratio is used for the evaluation results by simulation. As a result, both PI and SI achieved SPFM and LFM, which are the criteria of ASIL-B and ASIL-D, respectively.

VI. Conclusion

In this article, we show how we implemented the multicore SoC for automotive applications, presenting the SoC architecture to achieve high performance with low power consumption and functional safety approaches to achieve high safety.

The SoC has the heterogeneous architecture to achieve high performance with low power consumption. It consists of ten processors, four DSPs, and eight types of accelerators, including the DNN accelerator and the ISPs. These accelerators have power-aware features, such as parallel calculation units and local memory, for reducing the need for DRAM access.

Its SMs are introduced to support functional safety. Partitioning of the "PI" and "SI" achieves both high performance and high safety and can prevent fault propagation from PI to SI. Diagnostic features are introduced to detect several faults. Above all, a dedicated controller is introduced to realize runtime logic BIST for the ISP.

The SoC is implemented in a 16-nm process, and its size is 94.52 mm^2. The SoC achieved the performance of 20.5 TOPS and 2.1 TOPS/W.

References

[1] Euro NCAP. (2017). *Euro Ncap 2025 Roadmap: In Pursuit of Vision Zero*. [Online]. Available: https://cdn.euroncap.com/media/30700/euroncap-roadmap-2025-v4.pdf

[2] M. Daily, S. Medasani, R. Behringer, and M. Trivedi, "Self-driving cars," *Computer*, vol. 50, no. 12, pp. 18–23, Dec. 2017.

[3] S. Kato *et al.*, "Autoware on board: Enabling autonomous vehicles with embedded systems," in *Proc. ACM/IEEE 9th Int. Conf. Cyber-Phys. Syst. (ICCPS)*, Apr. 2018, pp. 287–296.

[4] H. Fan *et al.*, "Baidu Apollo EM motion planner," 2018, *arXiv:1807.08048*. [Online]. Available: https://arxiv.org/abs/1807.08048

[5] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst. (NIPS)*, Dec. 2012, pp. 1097–1105.

[6] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2014, *arXiv:1409.1556*. [Online]. Available: https://arxiv.org/abs/1409.1556

[7] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2014, pp. 580–587.

[8] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 3431–3440.

[9] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 770–778.

[10] *Taxonomy and Definitions for Terms Related to Driving Automation Systems for On-Road Motor Vehicles*. Standard SAE International J3016_201806, 2016. [Online]. Available: https://www.sae.org/standards/content/j3016_201806/
104

[11] Y. Tanabe *et al.*, "A 464GOPS 620GOPS/W heterogeneous multi-core SoC for image-recognition applications," in *IEEE ISSCC Dig. Tech. Papers*, Feb. 2012, pp. 222–223.

[12] J. Tanabe *et al.*, "18.2 A 1.9TOPS and 564GOPS/W heterogeneous multicore SoC with color-based object classification accelerator for image-recognition applications," in *IEEE ISSCC Dig. Tech. Papers*, Feb. 2015, pp. 328–329.

[13] Z. Nikolic, R. Venkatasubramanian, J. A. Jones, and P. Labaziewicz, "A scalable heterogeneous multicore architecture for ADAS," in *Proc. IEEE Hot Chips 27 Symp. (HCS)*, Aug. 2015, pp. 1–32.

[14] C. Takahashi *et al.*, "4.5 A 16nm FinFET heterogeneous nona-core SoC complying with ISO26262 ASIL-B: Achieving 10-7 random hardware failures per hour reliability," in *IEEE ISSCC Dig. Tech. Papers*, Jan./Feb. 2016, pp. 80–81.

[15] K. J. Lee *et al.*, "A 502-GOPS and 0.984-mW dual-mode intelligent ADAS SoC with real-time semiglobal matching and intention prediction for smart automotive black box system," *IEEE J. Solid-State Circuits*, vol. 52, no. 1, pp. 139–150, Jan. 2017.

[16] M. Ditty, A. Karandikar, and D. Reed, "Nvidia's xavier SoC," in *Proc. IEEE Hot Chips Symp. (HCS)*, Aug. 2018, pp. 1–17.

[17] *Road Vehicles-Functional Safety*, Standard 26262, 2018.

[18] A. Takeda, Y. Ishikawa, T. Mori, T. Kodaka, Y. Okuda, and T. Yoshikawa, "Hybrid adas development environment for architecture-specific performance estimation," in *Proc. 56th Annu. Design Autom. Conf. Designer Track*, Jun. 2019, pp. 1–11. [Online]. Available: http://www2.dac.com/56th/proceedings/slides/18_5.pdf

[19] T. Watanabe, S. Ito, and K. Yokoi, "Co-occurrence histograms of oriented gradients for pedestrian detection," in *Proc. Pacific–Rim Symp. Image Video Technol.* Berlin, Germany: Springer, Jan. 2009, pp. 37–47.

[20] G. Desoli *et al.*, "14.1 A 2.9TOPS/W deep convolutional neural network SoC in FD-SOI 28 nm for intelligent embedded systems," in *IEEE ISSCC Dig. Tech. Papers*, Feb. 2017, pp. 238–239.

[21] J. Lee, C. Kim, S. Kang, D. Shin, and S. Kim, "UNPU: A 50.6 TOPS/W unified deep neural network accelerator with 1b-to-16b fully-variable weight bit-precision," in *IEEE ISSCC Dig. Tech. Papers*, Feb. 2018, pp. 218–220.

[22] Y. Jia *et al.*, "Caffe: Convolutional architecture for fast feature embedding," in *Proc. 22nd Int. Conf. Multimedia*, 2014, pp. 675–678.

[23] S. Dabral, S. Kamath, V. Appia, M. Mody, B. Zhang, and U. Batur, "Trends in camera based automotive driver assistance systems (ADAS)," in *Proc. IEEE 57th Int. Midwest Symp. Circuits Syst. (MWSCAS)*, Aug. 2014, pp. 1110–1115.

Yutaka Yamada (M'19) received the B.E. and M.E. degrees from Keio University, Yokohama, Japan, in 2003 and 2005, respectively.

He joined Toshiba Corporation, Kawasaki, Japan, in 2005. He was involved in the research and development of reconfigurable processors and image processing accelerators. Since 2017, he has been a Research Engineer with Research & Development Center, Toshiba Electronic Devices & Storage Corporation, Kawasaki. His current research interests include the area of image recognition accelerators, image signal processors, and system architecture for automotive SoCs.

Mr. Yamada is also a member of the Institute of Electronics, Information and Communication Engineers (IEICE).

Toru Sano received the B.E. and M.E. degrees in information and computer science, under Prof. H. Amano, from Keio University, Yokohama, Japan, in 2007 and 2009, respectively.

He joined Toshiba Corporation, Kawasaki, Japan, in 2009. He is currently engaged in the research and development of automotive SoCs. His research interests include many-core processors, network-on-chip technology, reconfigurable processors, and image recognition/matching accelerators.

Yasuki Tanabe received the B.E., M.E., and Ph.D. degrees from Keio University, Yokohama, Japan, in 2002, 2004, and 2007, respectively.

He joined Toshiba Corporation, Kawasaki, Japan, in 2007, where he was involved in the research and development of image processing accelerators. From 2014 to 2016, he was a Visiting Research Scholar with Pennsylvania State University, State College, PA, USA, where he was involved in AI processing accelerator design. He is a Research Engineer with Research & Development Center, Toshiba Electronic Devices & Storage Corporation, Kawasaki. His current research interests include the area of image processing accelerators, image signal processors, and AI accelerators for automotive SoCs.

Yutaro Ishigaki (M'12) graduated from the Tokyo National College of Technology, Tokyo, Japan, in 2012. He received the B.S. and M.S. degrees in electrical and electronic engineering from the Tokyo University of Agriculture and Technology, Tokyo, in 2014 and 2016, respectively.

He joined Toshiba Corporation, Kawasaki, Japan, in 2016. He is currently working on developing image recognition processors at the Research & Development Center, Toshiba Electronic Devices & Storage Corporation, Kawasaki. His research interests include microprocessors, image processing, and hardware acceleration.

Mr. Ishigaki is also a member of the Institute of Electronics, Information and Communication Engineers (IEICE).

Soichiro Hosoda received the B.E. and M.E. degrees from Waseda University, Tokyo, Japan, in 2003 and 2005, respectively.

He joined Toshiba Corporation, Kawasaki, Japan, in 2005. He is currently with Research & Development Center, Toshiba Electronic Devices & Storage Corporation, Kawasaki. His current research interests include the areas of the interconnect architecture and image signal processors (ISPs) for automotive SoCs.

Fumihiko Hyuga received the B.E. and M.Sc. degrees from Kyoto University, Kyoto, Japan, in 2006 and 2008, respectively.

He joined Toshiba Corporation, Kawasaki, Japan, in 2008. Since 2011, he has been involved in the research and development of image processing accelerators and working as a research engineer at the Research & Development Center, Toshiba Electronic Devices & Storage Corporation, Kawasaki. His current research interests include the areas of image processing accelerators, image signal processors, and computer vision for automotive applications.

Akira Moriya received the B.E. and M.E. degrees from the Tokyo Institute of Technology, Tokyo, Japan, in 2007 and 2009, respectively.

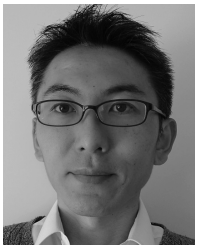
He joined Toshiba Corporation, Kawasaki, Japan, in 2009. He was involved in the research and development of image processing accelerators. Since 2017, he has been a Research Engineer with Research & Development Center, Toshiba Electronic Devices & Storage Corporation, Kawasaki. His current research interests include the area of image signal processors for automotive SoCs.

Ryuji Hada received the B.E. and M.E. degrees from Hiroshima City University, Hiroshima, Japan, in 2001 and 2007, respectively.

He joined Toshiba Corporation, Kawasaki, Japan, in 2007. He was involved in the research and development of image processing accelerators. Since 2017, he has been a Research Engineer with Research & Development Center, Toshiba Electronic Devices & Storage Corporation, Kawasaki. His current research interests include the areas of image processing accelerators, image signal processors, and the tool for automotive SoCs.

Atsushi Masuda received the B.E. and M.E. degrees from Tohoku University, Sendai, Japan, in 1986 and 1988, respectively.

He joined Toshiba Corporation, Kawasaki, Japan, in 1988. He was involved in the research and development of high-level synthesis. Since 2013, he has been a Research Engineer with Research & Development Center, Toshiba Electronic Devices & Storage Corporation, Kawasaki. His current research interest includes the area of image processing accelerators.



Masato Uchiyama received the M.S. degree in information and computer science from Keio University, Yokohama, Japan, in 2002, with a focus on processor and large-scale integrated circuits (LSI) architecture.

From 2002 to 2016, he worked on the hardware of media processors, multi-media CODECs, and image recognition at the Center for Research and Development, Toshiba Corporation, Kawasaki, Japan. Since 2017, he has been working on the development of an automotive SoC at the Electronic Devices & Storage Research & Development Center, Toshiba Electronic Devices & Storage Corporation, Kawasaki.

Masashi Jobashi is currently a System Large-Scale Integrated Circuits (LSI) Development Engineer with Toshiba Electronic Device & Storage Corporation, Kawasaki, Japan. He is also in charge of the implementation of image processing accelerators into the system for automotive SoCs and the evaluation of the functional performance of the SoC under the combinational operation of accelerators, CPUs, DSPs, videos, and so on.

Tomohiro Koizumi is currently a System Large-Scale Integrated Circuits (LSI) Development Engineer with Toshiba Electronic Device & Storage Corporation, Kawasaki, Japan.

Takanori Tamai received the B.E. and M.E. degrees from Waseda University, Tokyo, Japan, in 1996 and 1998, respectively.

He joined Toshiba Corporation, Kawasaki, Japan, in 1998. He was involved in the research and development of microprocessors and SoCs. Since 2019, he has been an Engineer with System Devices Division, Development Department, Micro Controller Unit, Toshiba Electronic Devices & Storage Corporation, Kawasaki. His current research interests include the areas of image signal processors and automotive electronics.

Nobuhiro Sato is currently a System Large-Scale Integrated Circuits (LSI) Development Engineer with Toshiba Electronic Device & Storage Corporation, Kawasaki, Japan.

Jun Tanabe received the B.S. degree in electrical engineering and the M.S. degree in computer science from Keio University, Yokohama, Japan, in 1998 and 2000, respectively.

In 2000, he joined Toshiba Corporation, Kawasaki, Japan. He is currently a Senior Specialist with Research & Development Center, where he has been working on image recognition SoCs.

Katsuyuki Kimura received the M.S. degree in information and computer science with a focus on processor and large-scale integrated circuits (LSI) architecture from Keio University, Yokohama, Japan, in 2002.

From 2002 to 2016, he worked on the hardware of media processors, multi-media CODECs, and image recognition at the Center for Research and Development, Toshiba Corporation, Kawasaki, Japan. Since 2017, he has been working on the development of an automotive SoC at the Electronic Devices & Storage Research & Development Center, Toshiba Electronic Devices & Storage Corporation, Kawasaki.

Yoshinari Ojima is currently a System Large-Scale Integrated Circuits (LSI) Development Engineer with Toshiba Electronic Device & Storage Corporation, Kawasaki, Japan.

Ryusuke Murakami received the B.E. degrees from Kyoto University, Kyoto, Japan, in 1986.

He joined Toshiba Corporation, Kawasaki, Japan, in 1986. He was involved in the development of voice synthesis large-scale integrated circuits (LSI), video decoder LSI, digital TV LSI, and automotive LSI. He is a Senior Expert with System Device Division, Toshiba Electronic Devices & Storage Corporation, Kawasaki.

Takashi Yoshikawa received the B.E. and M.E. degrees from the Tokyo Institute of Technology, Tokyo, Japan, in 1995 and 1997, respectively.

He joined Toshiba Corporation, Kawasaki, Japan, in 1997. He is currently a Senior Manager with Research & Development Center, Toshiba Electronic Devices & Storage Corporation, Kawasaki.