Received: 28 March 2020 | Revised: 15 June 2020 | Accepted: 2 July 2020

DOI: 10.4218/etrij.2020-0128

SPECIAL ISSUE

ETRI Journal WILEY

40-TFLOPS artificial intelligence processor with function-safe programmable many-cores for ISO26262 ASIL-D

Jinho Han[1] | **Minseok Choi**[1] | **Youngsu Kwon**[2]

[1]AI Processor Research Section, Electronics and Telecommunications Research Institute, Daejeon, Rep. of Korea

[2]AI SoC Research Division, Electronics and Telecommunications Research Institute, Daejeon, Rep. of Korea

Correspondence
Jinho Han, AI Processor Research Section, Electronics and Telecommunications Research Institute, Daejeon, Rep. of Korea.
Email: soc@etri.re.kr

Funding information
This research was supported by the Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No. 2018-0-00195, Artificial Intelligence Processor Research Laboratory).

The proposed AI processor architecture has high throughput for accelerating the neural network and reduces the external memory bandwidth required for processing the neural network. For achieving high throughput, the proposed super thread core (STC) includes 128 × 128 nano cores operating at the clock frequency of 1.2 GHz. The function-safe architecture is proposed for a fault-tolerance system such as an electronics system for autonomous cars. The general-purpose processor (GPP) core is integrated with STC for controlling the STC and processing the AI algorithm. It has a self-recovering cache and dynamic lockstep function. The function-safe design has proved the fault performance has ASIL D of ISO26262 standard fault tolerance levels. Therefore, the entire AI processor is fabricated via the 28-nm CMOS process as a prototype chip. Its peak computing performance is 40 TFLOPS at 1.2 GHz with the supply voltage of 1.1 V. The measured energy efficiency is 1.3 TOPS/W. A GPP for control with a function-safe design can have ISO26262 ASIL-D with the single-point fault-tolerance rate of 99.64%.

KEYWORDS
AI Processor, function-safe, ISO26262, many-core architecture

1 | INTRODUCTION

Artificial-intelligence (AI) processors are being researched to accelerate AI algorithms [1–3], which are being used for obstacle recognition and posture control in autonomous vehicles [4–7]. AI processors must be developed for autonomous vehicles [8–9]. Tesla's full self-driving (FSD) chip [10] is designed using dedicated hardware that can accelerate AI algorithms for performing object recognition in autonomous vehicles and functional-safety design for operation in an error-prone environment. Two FSD chips are used to process eight camera inputs, and each input identifies the surroundings and recognizes obstacles on the road. The performance of the proposed fault-tolerant feature is analyzed according to the ISO26262 standard, which is the standard to guarantee functional safety under unreasonable risks due to the hazards generated by the malfunctioning of the electrical system of a vehicle [11].

In automotive applications, the processor requires a fault-tolerant feature to prevent the transient faults due to voltage fluctuation, wide temperature variation, and exposure to particle radiation. Moreover, the advanced driver-assistant system (ADAS) processor installed in the vehicle should be extremely robust and stable in its operation to guarantee safety and convenience [12].

The functional safety of the processor with fault tolerance is analyzed according to the safety-element out-of-context (SEooC) of ISO26262. The SEooC is a safety

element developed out of context with respect to a specific vehicle-level application. It is intended to be used in multiple items when the validity of its assumptions can be established during the integration of the SeooC. The qualification of the software and hardware components addresses the use of the pre-existing elements for an item developed under ISO26262. The components are not necessarily designed for reusability or developed under ISO26262. The semiconductor has to be analyzed along with the vehicle system, and functional-safety requirement (FSR) and technical-safety requirement (TSR) have to be defined at the system level. because the processor cannot be analyzed along with vehicle-level applications, the functional safety is analyzed as SEooC with assumptions of ADAS system-level and FSR.

In failure mode effect analysis (FMEA), failure mode and the effects of the failure mode are observed. The failure mode is the manner in which an element or an item fails. It is found using a sub-module, and the effect is found using the fault ratio in the area. Therefore, the greater is the product of a single-point fault ratio and area, the more often failures are generated. We can reduce the effects of failure modes in the processor for ensuring the safety mechanism (SM) by using fault-monitoring systems analyzed via FMEA.

The hardware metric as a single-point fault metric (SPFM) is calculated using a base fault rate (BFR), which is calculated using IEC62380 standards [13], fraction of safe faults, and diagnostic coverage, which is the proportion of the hardware-element failure rate that is detected or controlled using the implemented safety mechanisms (SMs). We assume that there is no safe fault in the submodule. SPFM is low by this assumption. The diagnostic coverage is solved via a fault-injection experiment.

Additionally, ISO/PAS21448 [14] is used for the standardization of the functional safety of an autonomous system. ISO26262 standard addresses vehicle safety in the presence of unreasonable risks resulting from the malfunctions of the E/E system. Other parts of ISO26262 provide requirements and guidance to avoid and control the random hardware failures and systematic failures that could violate safety goals. For some systems, which rely on sensing the environment, there can be safety violations due to the limitations in the intended function of a system that is free from faults, as defined in ISO26262. Such safety violations include the inability of the function to correctly comprehend the situation and safely operate; The absence of this class of safety violations is defined as the safety of the intended functionality (SOTIF). The measures to reduce the risk of the SOTIF includes functions that use machine-learning algorithms, because of insufficient robustness of the system with respect to the sensor input variations, or diverse environmental conditions. The measures become important to address SOTIF in the design, verification, and validation phases.

The proposed AI processor architecture has two distinguishing features. First, it has high throughput for accelerating the neural network. Second, it reduces the required external memory data bandwidth for processing the neural network.

For achieving high throughput, the AI processor has a super thread core (STC) that includes 128 × 128 nano cores (NCs) operating at the clock frequency of 1.2 GHz. An NC is a processor core with an execution unit for processing CNNs, RNNs, and FCNs. In the STC, 16 384 NCs are integrated and operate at 1.2 GHz in a normal corner case.

Generally, the AI processor repeatedly reads the kernel and feature data of the neural networks from the external memory to process the neural networks. This means that a high data bandwidth for the external memory is required. Therefore, a unified memory for kernel and feature data is proposed to reduce the required external memory data bandwidth.

The proposed memory has the size of 40 MB with 3 × 256-bit AXI master and 1 × 256-bit AXI slave ports, as well as 256 read/write ports for the STC that operates at the clock frequency of 1.2 GHz. An outstanding feature of this memory is to store the feature and kernel data, which are read from the external memory. The stored data can be reused without accessing the external memory. This helps increase the operation intensity.

Additionally, using the power-gating function, the AI processor can power the 16 384 NCs that are operating. When the data and instructions are read and written at the unified memory, all the NCs can be power-off via power-gating. The unified memory is power-gated by 4 MB and has sleep mode by 32 KB to reduce the peak power consumption.

A function-safe architecture is proposed for a fault-tolerant system, such as the electronics system of an autonomous car. The general-purpose processor (GPP) core is integrated with the STC to control it and process the AI algorithm, and it has a self-recovering cache and dynamic lockstep (DLS) function.

DLS has a lockstep mode for the fault-tolerant function of controlling the STC and pre-processing and post-processing of the AI algorithm, and a dual-core mode for high performance. The self-recovering cache discovers the fault of the cache and recovers the fault by using the characteristics of the cache.

In this paper, we demonstrate that the fault performance of the function-safe architecture provides ISO26262 standard automotive safety integrity level (ASIL) D using the fault injection system proposed and adopted in the ISO26262 2nd Edition standard. The proposed architecture also provides superior fault tolerance compared with the conventional dual modular redundancy (DMR) and fault tolerant cache. In the absence of safety violations in this class complying with ISO/PAS21448, functional modifications to the design plans were used to reduce the SOTIF risk. We designed the

inter- and intra-frame safety measure, which is a type of SOTIF improvement measure for diverse sensor technology, and increased the performance of the recognition and decision algorithm by using functional restriction to mitigate the SOTIF risk. We then designed the functional-safety processor based on a deep-learning algorithm to process image data. The remainder of this paper is organized as follows: Section 2 details the proposed architecture; Section 3 explains its safety features; Section 4 provides a fault analysis that complies with ISO26262 and ISO/PAS21448, along with the implementation results; and the conclusions are presented in Section 5.

2 | AI PROCESSOR

2.1 | Many-core architecture

This processor includes an STC with 128 × 128 NCs, 40 MB unified kernel and feature memory, 2 GPP cores, 2 function-safe processor cores, a 16-lane PCIE Gen3, and a 2-channel LPDDR4 controller, as depicted in Figure 1.

To achieve high computational throughput for processing AI algorithms, the AI processor has an STC with a multi-core architecture. The STC comprises 128 × 128 processing cores called NCs [15]. The processing core operates at 1.2 GHz. The AI Processor includes a kernel and feature memory that reads the operand that must be operated on the processing core and an instruction memory that stores the instruction that must be operated on the processing core. Additionally, there exists a flow-control unit to control the flow of the kernel data, feature data, and instructions. Furthermore, the flow-control unit also reads instructions, feature data, and kernel data from the external memory through an on-chip bus and writes them to the feature memory, kernel memory, and instruction memory by using the direct memory access controller.

The maximum performance of 40 TFLOPS with half floating-point precision can be achieved using a 128 × 128 processing core, which can perform two 16-bit floating-point operations, namely, multiplication and accumulation, at 1 clock cycle as follows:

$$\begin{aligned} 2\,\text{Operations/cycle} \times 1.2\,\text{GHz} \times 128 \times 128 &= 32768\,\text{Giga FLOPS/sec} \\ &= 40\,\text{Tera FLOPS/sec} \end{aligned}$$

Additionally, the processing core NC is capable of the arithmetic for kernel operations, batch normalization, bias, scale, pooling, average pooling, and activation functions such as the rectified linear unit (ReLU), leaky ReLU, max pooling, load instruction, and reorganization layers, as shown in Figure 2. The processing core with 16 384 NCs was designed with a 20 000 μm × 15 000 μm die area using a 28-nm technology node. When accessing the feature memory and kernel

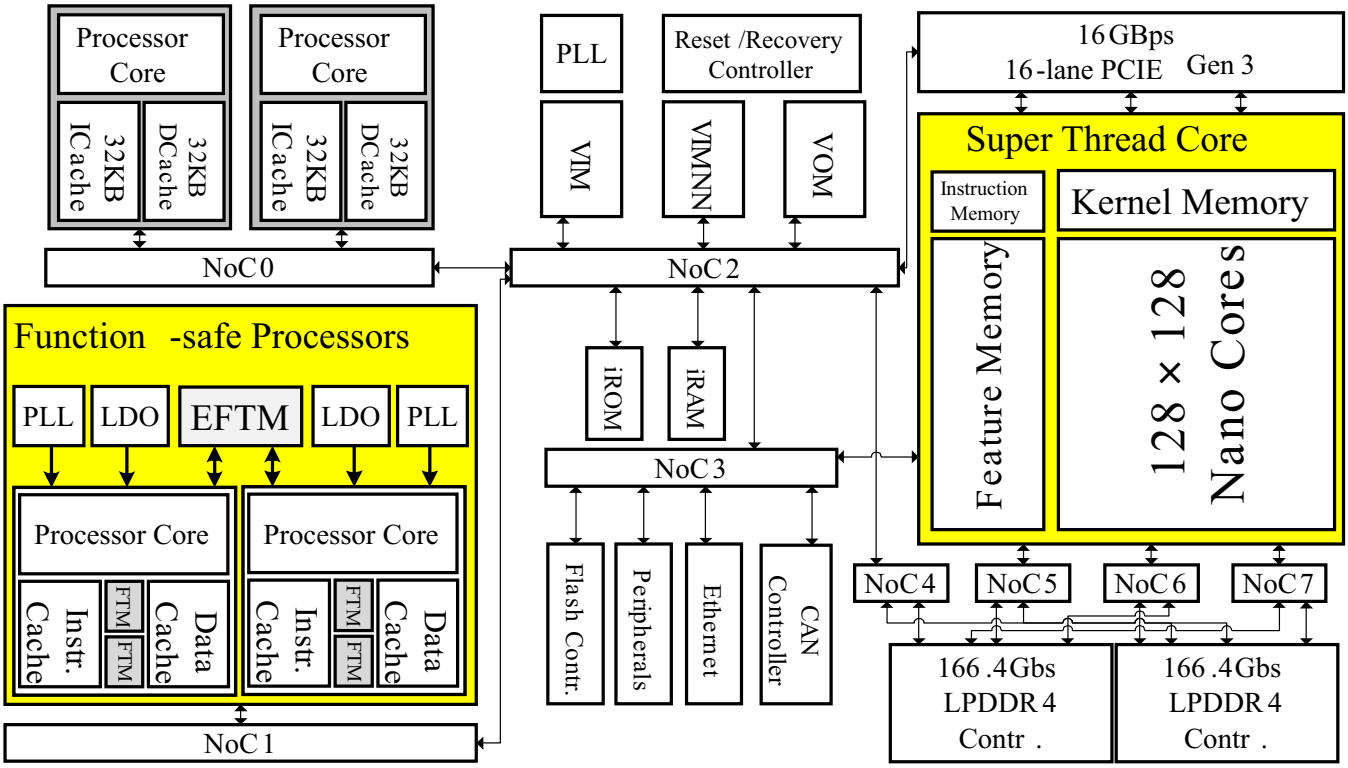
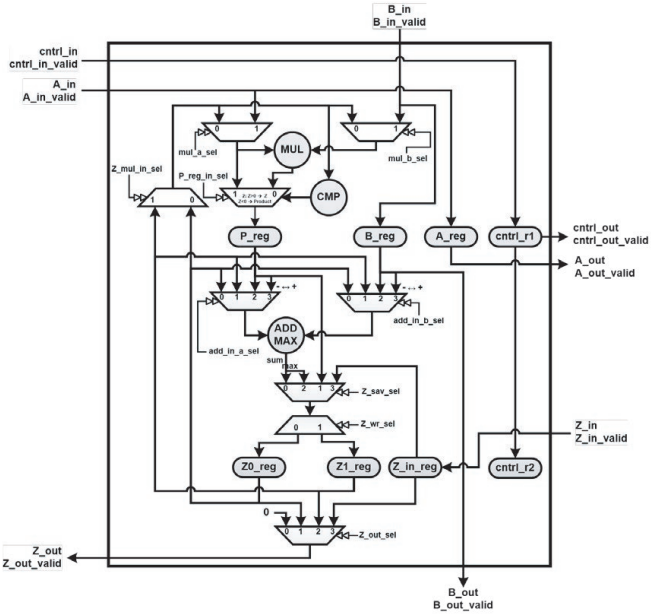


FIGURE 1 The overall architecture of AI processor



- **MUL** : ACC ß I * K + 0 (Kernel′s first)
- **MAC** : ACC ß I * K + ACC (Kernel′s rest)
- **ADD** : ACC ß I * K + ACC (Batch normalization, Bias)
- **MULK** : ACC ß ACC * K + 0 (Batch normalization, Scale)
- **MOVACC** : OUT ß ACC (Output, Pooling′s first)
- **OUTACC** : OUT ß OUT * 1 + ACC (Average Pooling)
- **MULOUT** : ACC ß OUT * K + 0 (Average Pooling)
- **CMPMUL0** : ACC ß ACC<0 ? ACC*0 : ACC (Activation- ReLU)
- **CMPMULK** : ACC ß ACC<0 ? ACC*K : ACC (Activation– Leaky ReLU)
- **MAX** : OUT ß Max(OUT, ACC) (Max. Pooling)
- **GETI** : ACC ß I * 1 + 0 (Load PartialSum)
- **PASS** : NOOP (Load PartialSum)
- **NOOP** : NOOP
- **SWAP** : ACC ß OUT, OUT ß ACC (Reorganization Layer)

FIGURE 2 Block diagram of nano core

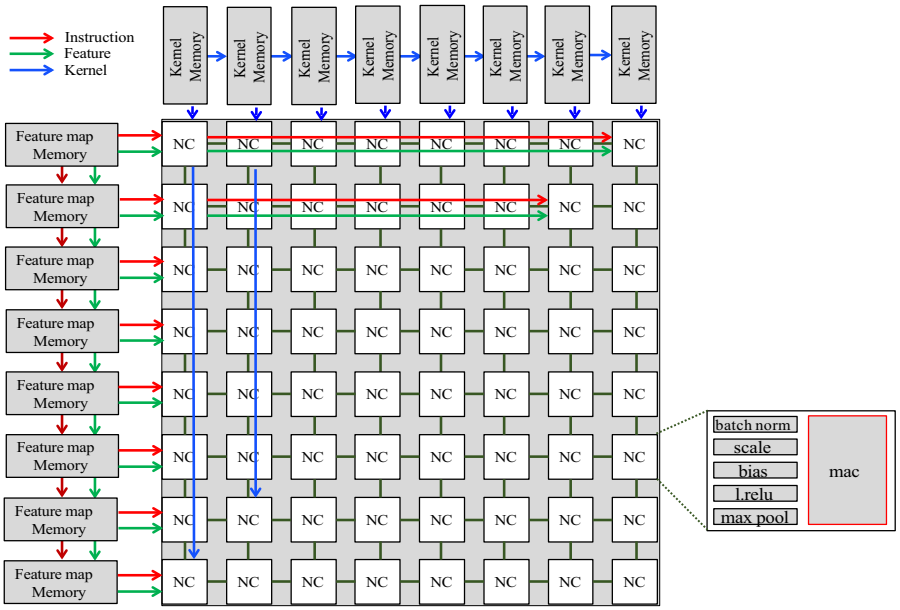


FIGURE 3 Block diagram of super thread core

memory from the processing core, the data and control signals can arrive at the desired destination with wire delays of up to 35 000 μm. The data and control signals also have long wire delays when accessing the feature memory and kernel memory from the flow-control unit. The feature memory, kernel memory, instruction memory, and processing core are accordingly connected using mesh topology with systolic data and instruction transfer to prevent performance degradation due to communication delays between the memories and processing cores.

2.2 | Programmability

To explain the operation of the processing cores, in Figure 3, we depict the AI processor architecture that comprises a 128 × 128 processing core.

The processing core requires two data and one instruction to operate. The two data become the feature data and kernel data, respectively. The memory near respective processing cores for these instructions and data do not exist; instead, they are supplied by the instruction memory, feature memory,

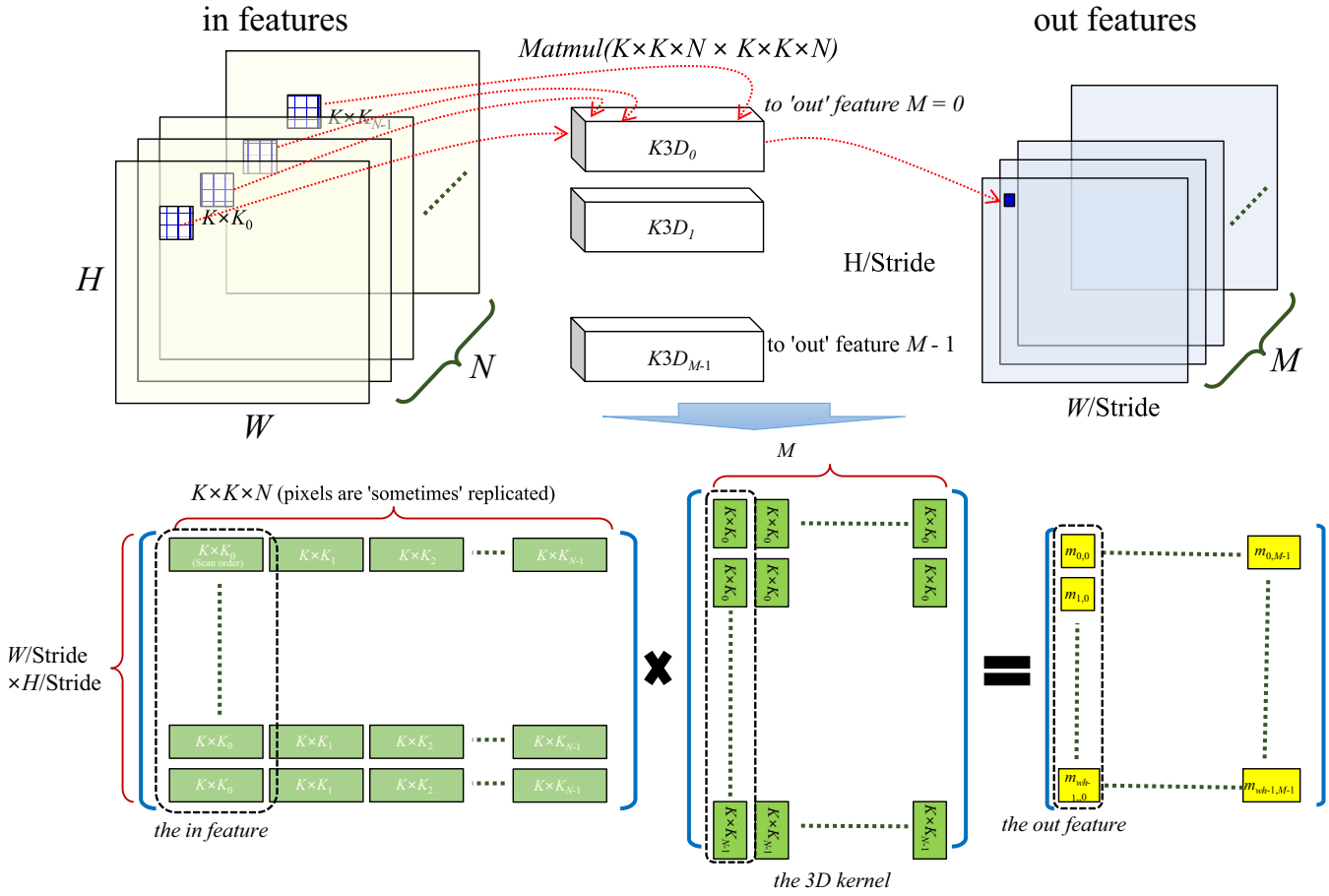


FIGURE 4 The specialized matrix for a convolution layer using many-core architecture

and kernel memory. The feature memory is on the left side of each row in the processing core array. The kernel memory is fed from the kernel memory at the top of each vertical row of the processing core array.

The supplied kernel data are delivered to the processing core below via the processing core at every clock cycle. Conversely, the adjacent processing cores in a vertical row receive the same kernel data sequentially but with a delay of one clock cycle. The feature data that are supplied are delivered to the processing core on the right through the processing core at every clock cycle. Conversely, the adjacent processing cores in a horizontal row receive the same feature data sequentially but with a delay of one clock cycle.

There exists only one instruction memory in the 128 × 128 STC. The instructions from the instruction memory are delivered first to the rightmost and topmost NC of the 128 × 128 STC, then subsequently delivered to both the lower and right NCs via the processing core at every clock cycle. Therefore, one instruction is delivered to all 128 × 128 NCs; however, the time at which the instruction is delivered to each NC is different. Additionally, the kernel data and feature data to be computed using the instructions are different; however, all the NCs execute the same instructions.

The convolution matrix operation typically performed in a CNN was restructured to enable efficient calculation using the proposed AI processor. As seen in Figure 4, the convolution matrix operation of an input feature map (IFM) of $H \times W \times N$ dimensions and an $M \times$ 3D kernel matrix ($K3D_0$ to $K3D_{M-1}$) of $K \times K \times N$ dimensions yields an output feature map (OFM) of $W/$ step $\times$ $H/$step $\times$ M dimensions. One pixel of the output feature map is the result of a convolution operation of $K \times K \times N$ pixels of the input feature map and one of the M kernels. Thus, the convolution operation is repeated within the dimensions of the OFM.

In this specialized matrix operation, the product of the $\{W/\text{step} \times H/\text{step}\} \times \{K \times K \times N\}$ matrix of repeated IFM pixels and the $\{K \times K \times N\} \times M$ matrix of $K3D_0$ through $K3D_{M-1}$ is equal to $\{W/\text{step} \times H/\text{step}\} \times M$ results in the OFM matrix.

The product of one row of the matrix $\{W/\text{step} \times H/\text{step}\} \times \{K \times K \times N\}$ and one column of the matrix $K \times K \times N\} \times M$ is equal to $\{W/\text{step} \times H/\text{step}\} \times M$ and becomes one element of the OFM matrix.

One row of the matrix $\{W/\text{step} \times H/\text{step}\} \times \{K \times K \times N\}$ becomes a vector consisting of $K \times K \times N$ elements, and the number of vectors is the number of OFM elements that can be obtained with one kernel.

3 | FAULT-TOLERANCE FEATURE

3.1 | General-purpose processor

The STC comprises 16 384 NCs. It is a significant increase in area due to redundant cores to take many of these cores to the traditional fault tolerant Architecture, a DMR-based structure. Therefore, considering that 16 384 NCs are not always used when executing the neural network algorithm, the error-tolerance function was implemented using software as a method of performing DMR. If the performance does not decrease, two groups of NCs run the same AI algorithm for a DMR feature. Additionally, this software is executed in the GPP, so that the GPP performs the error-tolerance function.

With fluctuating voltage, widening operating temperature, and increasing clock frequency, chip multiprocessors (CMPs) are becoming increasingly susceptible to transient errors, hard errors, manufacturing defects, and process variations. For performing error detection and recovery, DMR-based CMPs run programs with the redundant execution at the other core [16]. Additionally, this technique has been developed by the following advances: (a) The combination of redundant execution with simultaneous multi-threading (SMT) and a recovery function [17]. The recent shift toward more on-chip thread contexts has resulted in the development of a large core with threading support. (b) The addition of control logic for comparison of the lead thread with the trailing thread. (c) Compensating for the vulnerability to transient errors in the cache [18,19]. In our proposed approach, the cache system, which takes up two-thirds of the processor area, is a vulnerable block in the processor with PVT variations. Therefore, the cache system requires a fault-tolerance feature.

To design a fault-tolerant cache system, the error correction code (ECC) is an attractive approach for transient-error detection and correction [20]. However, the redundant memory for ECC significantly affects the cost and can increase the transient-error rates by increasing the die size, because of including the memory for the error code [21,22].

We developed a GPP. The processor has 13 pipeline stages with a two-issue superscalar architecture, a fetch scheduler that can fetch eight instructions at the maximum, a branch predictor with a branch target buffer that uses the GSHARE method with branch-history registers, a load-and-store unit with two pipeline stages, an I/D cache with three pipeline stages, and an I/D table look-aside buffer [23]. It operates at the frequency of 1.2 GHz on a 28-nm node to provide sufficient computing power for AI applications. In the processor, the cache system takes up to 70% of the processor area; therefore, it needs to be designed with fault-tolerant features to prevent the changing of the memory contents in the cache because of the transient error generated due to voltage drop and high temperature. In this study, we present a fault-tolerant cache system for processors [24,25]. The cache system has a smaller redundant memory than that for the ECC presented in [22], thereby decreasing the transient-error rates upon using the proposed mechanism, which increases the error-recovery rate. The proposed mechanism is considered with data cache characteristics and reconfigures the memory in the cache to prevent serious and permanent faults without the address-interleaving technique used in [21].

Additionally, we present the processor that has DMR with separate clock and power sources that work with a cache, which includes a self-recovering function and error prediction. Separate clock and power sources prevent dependent failures of two or more circuits resulting from a single specific event or root cause. The DMR and the self-recovering function detect the transient faults that occur once and subsequently disappear. Performing error prediction prevents permanent faults that occur and remain until they are removed or repaired.

The AI platform including the GPP with the fault-tolerance feature, overcomes the low fault coverage of the self-test and the performance overhead incurred due to the self-test time. An analysis showed that the proposed fault-tolerant processor complies with ISO26262. The AI processor SoC includes two cores with 32 KB caches, and functional safety using a DLS and self-recovering cache. The processor works in tandem with the 128 × 128 processing cores. [26]

The proposed fault-tolerant processor contains two key features: (a) DLS with separate clock and power sources to reduce dependent failures or achieve high performance and (b) a cache with the self-recovering function to reduce the instances of transient faults and a reconfigurable function to reduce the instances of permanent faults.

Nowadays, the semiconductors for automotive applications must comply with the ISO26262 standard to achieve functional safety. The ISO26262 standard guarantees functional safety, eliminating the unreasonable risk due to the hazards generated by malfunctioning of the electrical system of a vehicle. Radiation, noise, and variation in voltage, current, and temperature can induce a fault. The fault results in the failure of the logic and flip-flops in the semiconductor. The processor and the cache preventing from automotive malfunctions and hazards for vehicles are needed.

3.2 | Dynamic lockstep

The design operates in one of the following two modes: the DLS mode and non-DLS mode. The mode of operation is determined via software programming, as depicted in Figure 5. The processors operate in the DLS mode when the DLS registers of both the processors are enabled by the software, or they operate in the non-DLS mode when the DLS registers of

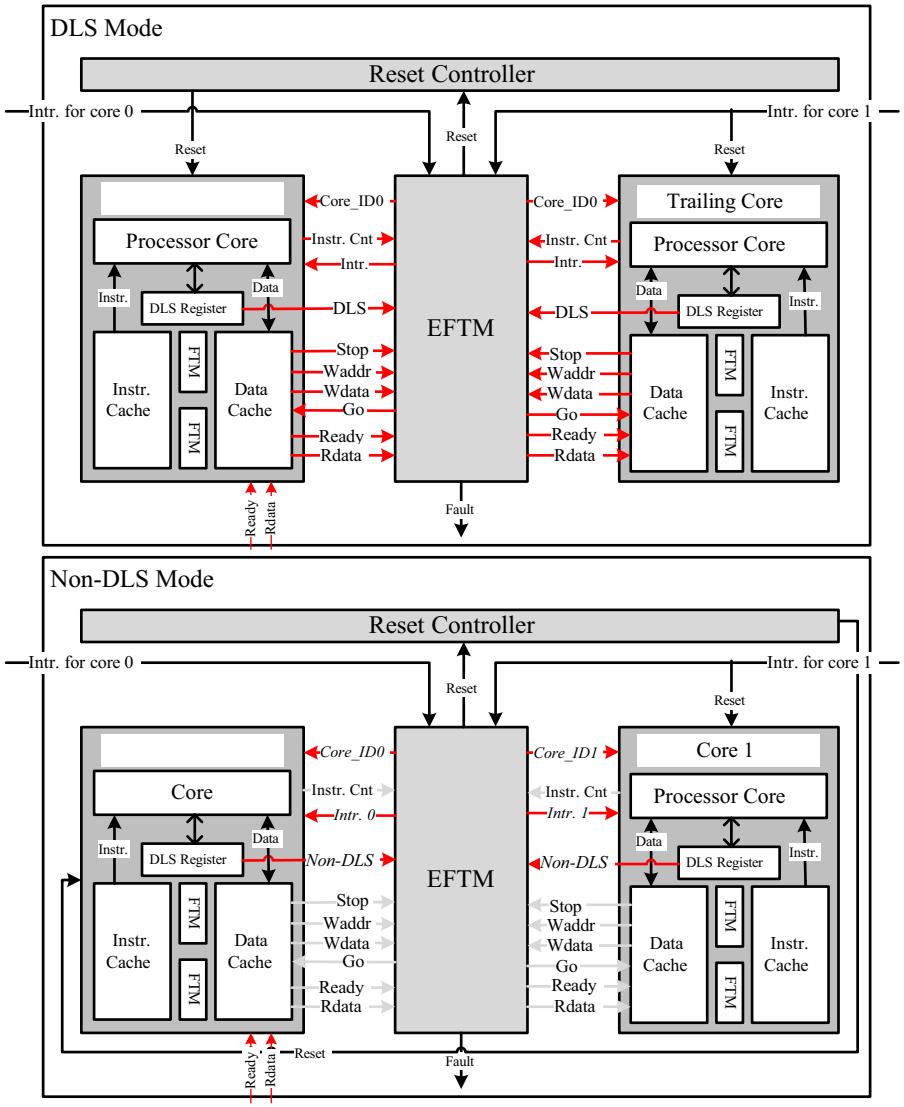


FIGURE 5 DLS Mode and Non-DLS Mode of DLS

both the processors are disabled by the software and an error predictor.

In the DLS mode, one processor operates as the leading core, while the other operates as a trailing core. Additionally, the operating frequency of the leading core is higher than that of the trailing core. The difference between the operating frequencies results in temporary redundancy, as depicted in Figure 3.

Both the processors perform the same task by controlling their core IDs using the EFTM. Notably, the EFTM stops the data caches when the processor core requests a write to the data cache and starts the data caches of the leading and trailing core if the cache write of the trailing core resembles that of the leading core. However, the EFTM generates a fault signal if it does not resemble. The data in the SDRAM are not changed by the trailing core because they are written in the data cache; however, the dirty bit of the data cache in the trailing core is not enabled.

Conversely, the data are changed by the leading core, and the trailing core can only read the data. The frequency of the check point that uses the cache write is sufficient to identify the fault of the processor. Additionally, to maintain the leading and trailing cores in the same state, the EFTM receives the interrupt for core 0 and sends it to the cores in the same processor state using the fetched instruction count of each processor. The state of the peripheral is not changed by the duplicated access, and only the leading core can read peripherals and transfer the data to the trailing core.

When a non-DLS mode value is stored in the DLS register, the EFTM sets the cores to the non-DLS mode. The interrupt signal of each core is transmitted to each core as it is. Additionally, different core IDs are assigned, so that different software can be executed.

In the non-DLS mode, both the processors operate as a traditional dual-core processor would; therefore, they increase the throughput by dynamically running different tasks on each processor, as depicted in Figure 3.

3.3 | Self-recovering cache

The processor detects transient faults and recovers from faults in the cache using the FTM. FTM reconfigures the processor to be resilient against permanent faults and restarts the processor to recover and avoid permanent faults by 1) running the recovery mechanism that recovers the data in caches using the lower-level cache and SDRAM, as well as the characteristics of the cache or resets the system; and 2) monitoring the load and store units of both the processor cores and detects faults by comparing the cache writes of both the processor cores and recovers using the recovery module as [25].

The ISO26262-compliant semiconductor has a fault-tolerant design for permanent faults. The design includes wearout prevention and built-in self-test; the multicore lockstep and part-wise checker, which is for transient faults; and the fault-tolerant design for dependent failures. The fault-tolerant designs for transient, permanent, and dependent failures are analyzed and verified via FMEA, fault tree analysis, qualitative analysis, and fault injection.

To comply with the ISO26262 standard, we defined the hazard analysis and risk assessment, which identifies hazards and hazardous events that must be prevented in the vehicle system including the proposed semiconductor, and the safety goal formulation for each hazardous event and ASIL for each safety goal. The satisfaction of FSR and TSR for the vehicle system according to the ASIL was proven by performing an analysis according to ISO26262-10.

3.4 | Intra-frame, inter-frame feature

The functional-safety mechanism for 128 × 128 processing cores in a GPP is depicted in Figure 6.

The intra-frame safety mechanism requires processing the same frame twice for a DMR using the batch mode in STC to detect the fault in the processing result of the frame. The frame_0_s0 is the batch with the input feature set of the leading frame. Additionally, the frame_0_s1 is the batch with the input feature set of the trailing frame, whose number is the same as that of the leading frame. The kernels of the leading frame and the trailing frame are the same if the number of the layer is the same. The result calculating yolov2 about the leading frame is compared with the one about the trailing frame. If the result about the leading frame is different than the one about the trailing frame. To judge whether the difference of the result is the resilient fault, the inter-frame safety mechanism is to be run.

The inter-frame safety mechanism is to compare the results of the current frame with those of the previous frame by using the SOTIF policy when the result of the leading frame is different than that of the trailing frame. when the difference between the previous frame result and the current frame result is smaller than the pre-defined threshold value. The result of the frame with a smaller difference between the leading and trailing frames is used. However, when the difference between the previous frame result and the current frame result is greater than the threshold value. The result of the current frame is not used. The threshold value is calculated using the rational distance and speed between the car and the obstacle every frame. The maximum number of the recognized objects is compared with the recognition percentage.

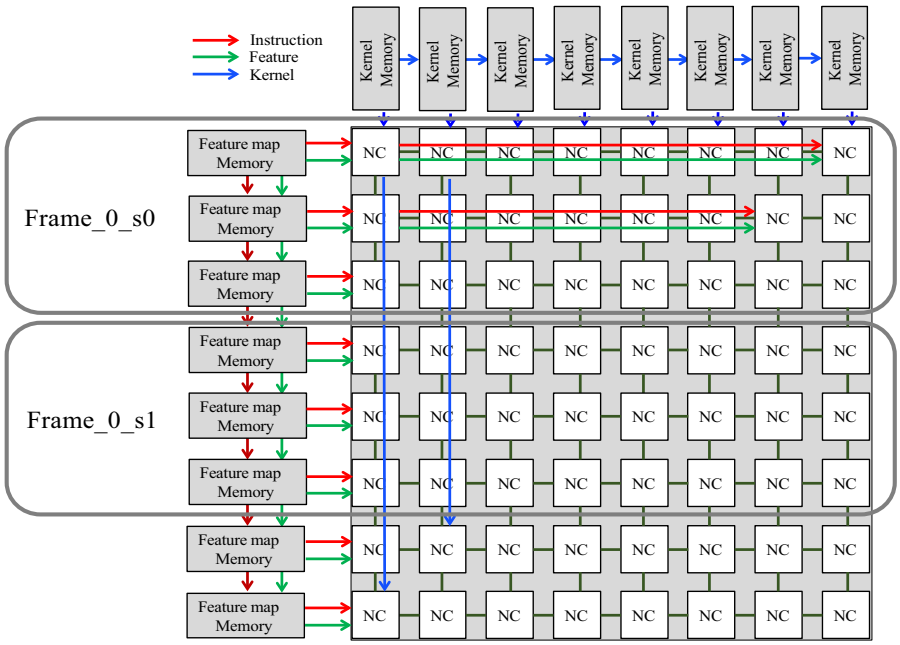
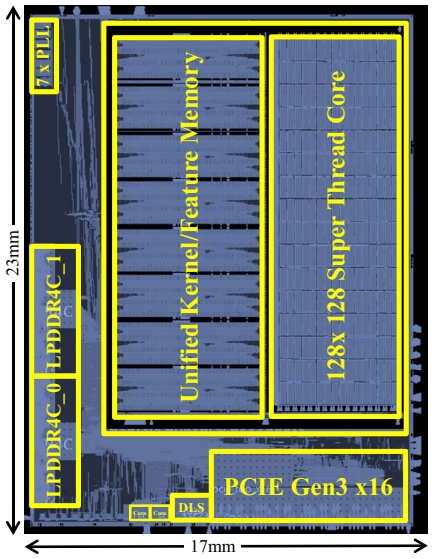


FIGURE 6 Block diagram of intra-frame safety



AI Processor with Function-safety, 2020	
Technology	28nm 1P12M Logic CMOS
Chip Size	391 mm^2
Gate Count	1 Billion gates
Supply Voltage	1.05 V—1.15 V
Clock Frequency	600 MHz–1.2 GHz
AI Performance	40 TFLOPS
GP Processor Core	2 × 1.2 GHz Processor Core 2 × 1.2 GHz DLS Core
Super Thread Core	128 × 128 1.2 GHz Nano Cores
STC Size	310 mm^2
Feature/Kernel Memory Size	40 MB SRAM (+8 MB Redundancy)
External Memory Interface	333 Gb/s 2-Channel LPDDR4
External Interface	16 GB/s 16-Lane PCIE Gen3
AI SDE	Neural Net Compiler, Simulator

FIGURE 7 Chip footprint

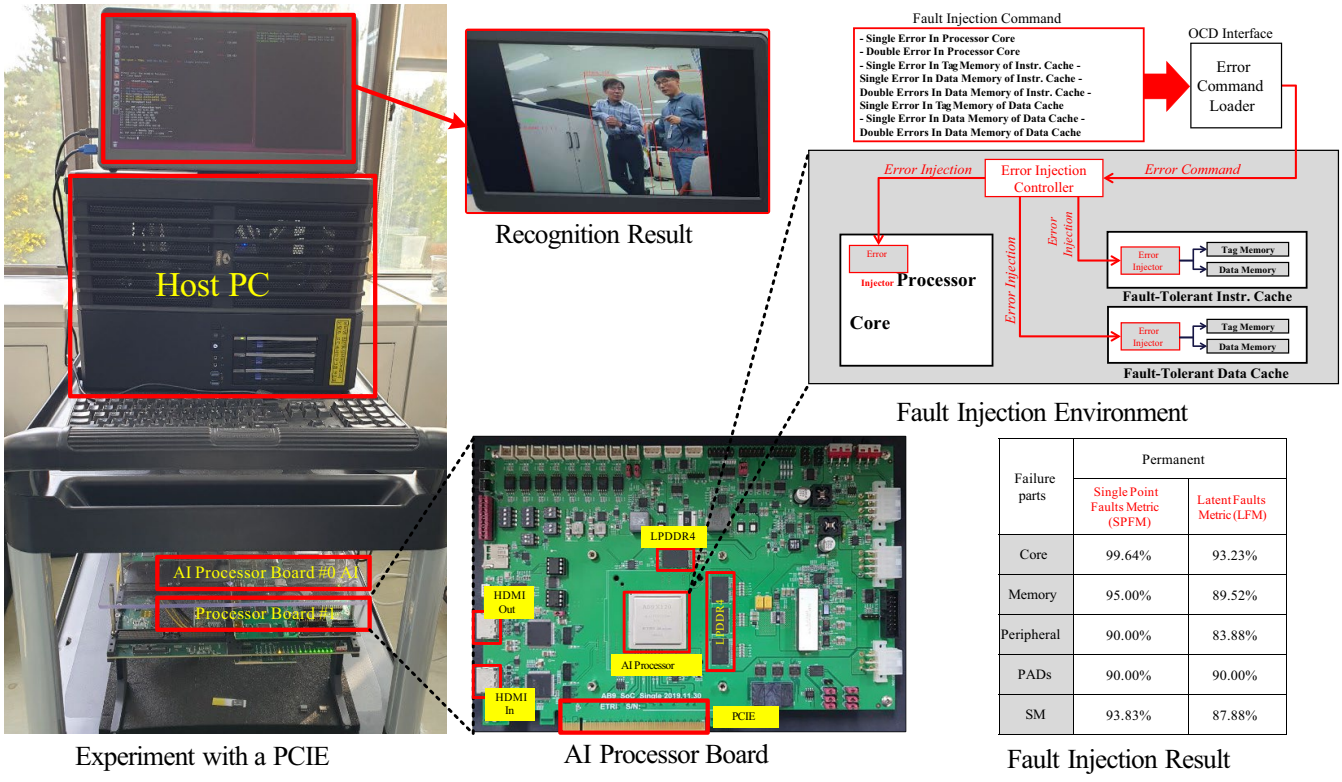


FIGURE 8 Experiment environment with fault injections

4 | VERIFICATION

4.1 | Chip implementation

The proposed AI processor is depicted in Figure 7. The AB9 chip photograph occupies 494 mm^2 in a 28 nm 12 metal CMOS technology with two input-NAND equivalent gate count of 1 B and 40 MB of the on-chip SRAM. This chip includes an STC with 128 × 128 NCs, 40 MB unified kernel and feature memory, 2 GPP cores, 2 function-safe processor cores, a 16-lane PCIE Gen3, and a 2-channel LPDDR4 controller. The chip specification is summarized. The AI processor has the peak power of 30 W with 1.05 V to –1.15 V supply voltage and 600 MHz to 1.2 GHz operating frequency. For the peak performance of 40 TFLOPS, the AI processor achieves the energy efficiency of 1.3 TOPS/W and area efficiency of 65.8 GOPS/mm^2 in the target applications.

Neural Network	Yolo v2	Yolo v3	MLP	LSTM
#Parameters	50.7M	160M	-	-
#MAC	17.5B	63B	-	-
Fps	150	75	-	-
STC(%)	32.8	42.8	81	14

The performance of Neural Network Computation with a Function-safe

Feature	TPU version 1	The proposed architecture
Multiplication	256x256OPs/cycle (8-bit INT) x 700MHz	128x128FLOPs/cycle (16-bit FP) x 1.2GHz
Addition	256x256OPs/cycle (8-bit INT) x 700MHz	128x128FLOPs/cycle (16-bit FP) x 1.2GHz
Accumulator	256 OPS/cycle x 700MHz	-
Activation	256 OPS/cycle x 700MHz	128 FLOPSs/cycle x 1.2GHz
Normalize/Pool	256 OPS/cycle x 700MHz	128x128 FLOPS/cycle x 1.2GHz
Bandwidth from buffer to MMU	167GB/s	128x2Bx1.2G/s(307GB/s)
Host Interface	14GiB/s	16GT/s
Bandwidth of Weight FIFO	30GiB/s	128x2Bx1.2GHz(307GB/s)
Bandwidth between Host Interface and buffer	10GiB/s	700MHzx256-bit/channel/cycle(22.4GB/s)
DDR bandwidth	30GB/s	1.3Gbx2x128-bit/s(41.6GB/s)

The feature comparison via google TPU version 1

	FTCache[18]	DCC[16]	Takahashi[12]	Renesas[27]	TI[28]	This work
Fault Detection	ECC	DMR Contr. by SW	Self-BIST	LSDC (lock-step dual-core), BIST	PGD, Separated clock & Power	DLS On DLA SW, Separated clock & Power
ECC	SECDED	N/A	N/A	N/A	N/A	SECDED
Fault Prediction &Recovery	X	X	Droop Monitor Adaptive Clock Control	X	PGD(power-glitch detect)	Self-recovering Cache, EFTM
Fault Injection	X	X	X	X	X	O
AI Engine	X	X	X	X	14TOPS MAC	40TFLOPS MAC
Fault Traps	83	N/A	10-7 RHF/Hour	N/A	N/A	28%
SPFM	N/A	N/A	>98%	N/A	>99%	99.64%
LFM	N/A	N/A	>60%	N/A	>90%	93.23%

The comparison of fault tolerance feature

FIGURE 9 Performance of AI processor

4.2 | Experiment board

The SoC board includes an AI processor, the 16-lane PCIE gen3 interface, four components of 8GB LPDDR4, video input chip, and video output chip, as depicted in Figure 8. The board receives images through a camera, recognizes them, and outputs them to the monitor. The program and kernel data, created using the NN compiler, are downloaded using an on-chip debugger.

The STC file, which contains neural-network description and weight information, and the CFG file, which contains the hardware settings, are created using the NN compiler. The application program, such as darknet.c with STC file and CFG file, is compiled using a GPP compiler and downloaded to the DRAM. Subsequently, the GPP is run using the on-chip debugger. The application then accepts the camera input and outputs the camera input to the monitor. Subsequently, the STC instruction, feature data, and kernel data are moved to the instruction memory, feature memory, and kernel memory in the STC, and yolov2 is run. The execution results are displayed on the monitor with object localization, object class, and confidence. Yolov2 was performed with the performance of approximately 68 fps. A 1280 × 720 video input is received through a camera (Sony) and stored in the DDR4 with 416 × 416 image size by the internal video input module. Additionally, the AI processors communicate using the PCIE interface and controlled and verified through the host PC.

By using PCIE, two boards that contain one AI processor each can communicate, and, therefore, the Yolov2 algorithm can be divided and executed. In the figure, some layers of Yolov2 are executed on AI Processor Board # 0, and others are executed on AI Processor Board # 1. The recognition result shows that the object class and confidence appear on the monitor to show the recognition result of the host PC.

4.3 | Performance complying with ISO26262

On the basis of fault analysis that complied with ISO26262 as a SEooC, we made a safety manual, including FMEA, hardware metrics with permanent faults, and fault coverage via fault injection based on TSR, and an SM.

Our SM is the fault-monitor system. We can find the fault rate by using the fault injector when the SM is applied. As shown in Figure 6, the single-point fault metric for permanent faults is 99.64%, and the latent-fault metric for permanent faults is 93.23%, with our SM including the fault-monitor systems. Our design can have ASIL D of ISO26262 in which the SPFM is more than 90%.

The performances of the neural networks yolov2, yolov3, MLP, and LSTM are shown in Figure 9. Because the conv1 layer of yolov2 has a 208 × 208 × 32 output feature matrix, all 128 × 128 NCs are activated simultaneously. However, because the 30th layer of yolov2 has a 13 × 13 × 425 output feature matrix, all 128 × 128 NCs are not activated simultaneously, so the effective performance of yolov2 is 32.8% that of 40 TFLOPS.

The proposed architecture is compared with the Google TPU v1 architecture in Figure 9. The proposed 16-bit floating-point structure that operates at 40 TFLOPS has 154 GFLOPS of activation operators and normalize/pool operators. The data bandwidth from the internal memory to the NCs is 307 GB/s, which is the same when providing weight. The data bandwidth between the host interface and the internal memory is 22.4GB/s. Finally, the two channel LPDDR4 has an external memory bandwidth of 41.6GB/s. Compared to Google's TPU version 1, the internal communication bandwidth of the proposed processor is higher, as is its peak performance when the 16-bit floating-point operation is four times the 8-bit integer operation.

Transient faults are analyzed using fault injection on the SoC board. The proposed fault-tolerance architecture has a fault-tolerant performance with a high fault coverage and low overhead in comparison with previous works [12,16,18,27,28], as depicted in Figure 9.

5 | CONCLUSION

We proposed an AI processor that adopts an STC and unified kernel/feature memory to increase the throughput while increasing its data reusability. Additionally, the AI processor includes a function-safe design with a self-recovering cache and DLS, and it runs the intra-fame, inter-frame function for the functional safety of STC.

The AI processor is fabricated via 28-nm CMOS process as a prototype chip and successfully verified on an FPGA board. The chip has the die area of 498 mm^2 and contains 1 B logic gates and a 40-MB on-chip SRAM. Its peak computing performance is 40 TFLOPS at 1.2 GHz at the supply voltage of 1.1 V, and the energy efficiency is 1.3TOPS/W. The proposed AI processor uses 64% of its NCs on average with a 40-MB unified memory for storing kernel and feature data, and has an effective performance of 20.5 TFLOPS.

Compared with Google TPU, it can take a large memory that can store both kernel data and feature data, so it can increase the effective performance when MLP and LSTM are performed. The LSTM operation was performed simultaneously by reading a lot of LSTM's input feature data with 40 MB of memory. Operation intensity can be increased by increasing data reusability, thus reducing external memory bandwidth. at the same time, a GPP for control with a function-safe design can have a 99.64% single-point fault-tolerance rate.

ORCID

Jinho Han https://orcid.org/0000-0002-0655-320X

REFERENCES

1. T. Luo et al., *DaDianNao: a neural network supercomputer*, IEEE Trans. Comput. **66** (2017), no. 1, 73–88.
2. N. Jouppi et al., *In-datacenter performance analysis of a tensor processing unit*, in Proc. ACM/IEEE Annu. Int. Sym. Comput. Architecture (Toronto, Canada), June 2017, pp. 1–12.
3. A. Parashar et al., *SCNN: An accelerator for compressed-sparse convolutional neural networks*, in Proc. ACM/IEEE Annu. Int. Symp. Comput. Architecture (Toronto, Canada), June 2017, pp. 27–40.
4. K. Simonyan and A. Zisserman, *Very deep convolutional networks for large-scale image recognition*, in Proc. Int. Conf. Learn. Representations (San Diego, CA, US), May 2015.
5. A. Krizhevsky et al., *ImageNet classification with deep convolutional neural networks*, Adv. Neural Inf. Process. Syst. **25** (2012), 1106–1114.
6. J. Redmon et al., *You only look once: unified, real-time object detection*, in Proc. IEEE Conf. Comput. Vision Pattern Recogn. (Las Vegas, NV, USA), 2016, pp. 779–788.
7. C. Szegedy et al., *Going deeper with convolutions*, ArXiv:1409.4842, 17th Sep 2014.
8. P. Gupta, *An Overview of NVIDIA's Autonomous Vehicles Platform*, in Proc. HotChips (Cupertino, CA, USA), 2017.
9. J. Choquette, O. Giroux, and D. Foley, *Volta: Performance and Programmability*, IEEE Micro **38** (2018), no. 2, 42–52.
10. P. Bannon et al., *Compute and redundancy solution for the full self-driving computer*, in Proc. 31th Hot Chips (Silicon Valley, CA, USA), 2019.
11. ISO26262 2nd Edition: Road vehicles – Functional Safety, 2018.
12. C. Takahashi et al., *A 16nm FinFET Heterogeneous Nona-Core SoC Complying with ISO26262 ASIL-B: Achieving 10–7 Random Hardware Failures per Hour Reliability*, in Proc. IEEE Int. Solid-State Circuits Conf. (San Francisco, CA, USA), 2016, pp. 80–81.
13. Reliability data handbook, IEC TR 62380, 2004.
14. ISO/PAS21448: Road vehicles - Safety of The Intended Functionality, 2019.
15. Y. Kwon et al., *Function-Safe Vehicle AI Processor with Nano Core-in-Memory Architecture*, in Proc. IEEE Int. Conf. Artif. Intell. Circuits Syst. (Hsinchu, Taiwan), Mar. 2019, 127–131.
16. A. Golander et al., *Synchronizing Redundant Cores in a Dynamic DMR Multicore Architecture*, IEEE Trans. Circuits Syst. II Exp. Briefs **56** (2009), no. 6, 474–478.
17. E. Rotenberg, *AR-SMT: A Microarchitectural Approach to Fault Tolerance in Microprocessors*, in Proc. Int. Symp. Fault-Tolerant Comput. (Madison, WI, USA), June 1999, pp. 84–91.
18. M. Zhang et al., *Reliable ultra-low-voltage cache design for many-core systems*, IEEE Trans. Circuits Syst. II Exp. Briefs **59** (2010), no. 12, 858–862.
19. M. R. Kakoee et al., *Variation-tolerant architecture for ultra low power shared-L1 processor clusters*, IEEE Trans. Circuits Syst. II Exp. Briefs **59** (2012), no. 12, 927–931.
20. A. R. Alameldeen et al., *Energy-efficient cache design using variable-strength error-correcting codes*, in Proc. Annu. Int. Symp. Comput. Architecture (San Jose, CA, USA), June 2011, pp. 461–471.
21. D. Rossi et al., *Error correcting code analysis for cache memory high reliability and performance*, in Proc. Design, Autom. Test Eur. (Grenoble, France), Mar 2011, pp. 1620–1625.
22. A. Neale et al., *Adjacent-MBU-Tolerant SEC-DED-TAEC-yAED Codes for Embedded SRAMs*, IEEE Trans. Circuits Syst. II Exp. Briefs **62** (2015), no. 4, 387–391.
23. Y. Kwon et al., *80mW/MHz 0.68V ultra low-power variation-tolerant superscalar dual-core processor*, IEIE Trans. Smart Process. Comput. **4** (2015), no. 2, 71–77.
24. J. Han et al., *80μW/MHz, 850MHz fault tolerant processor with fault monitor systems*, J. Semiconductor Technol. Sci. **17** (2017), no. 5, 627–635.
25. J. Han et al., *A fault tolerant cache system of automotive vision processor complying with ISO26262*, IEEE Trans. Circuits Syst. II: Express Briefs **63** (2016), no. 12, 1146–1150.
26. J. Han, Y. Kwon, and H.-J. Yoo, *A 1GHz fault tolerant processor with dynamic lockstep and self-recovering cache for ADAS SoC complying with ISO26262 in automotive electronics*, in Proc. IEEE Asian Solid State Circuits Conf. (Seoul, Rep. of Korea), Nov. 2017, pp. 313–316.
27. H. Kimura et al., *A 40 nm flash microcontroller with 0.80μs field-oriented-control intelligent motor timer and functional safety*

system for next-generation EV/HEV, in Proc. IEEE Int. Solid-State Circuits Conf. (San Francisco, CA, USA), Feb. 2017, pp. 58–59.
28. R. Venkatasubramanian et al., *A 16 nm 3.5B+ transistor >14TOPS 2-to-10W multicore SoC platform for automotive and embedded applications with integrated safety MCU, 512b vector VLIW DSP, embedded vision and imaging acceleration*, in Proc. IEEE Int. Solid-State Circuits Conf. (San Francisco, CA, USA), Feb. 2020, pp. 52–54.

AUTHOR BIOGRAPHIES

Jinho Han received B.S., M.S., and Ph.D. degrees from the Korea Advanced Institute of Science and Technology (KAIST), Republic of Korea in 1998, 2001, and 2020, respectively. He has been with the AI Processor Research Section of the Electronics and Telecommunications Research Institute (ETRI), Republic of Korea, since 2001. In ETRI, he is a Section Leader, Principal Research Staff of the AI Processor Research Section devoted to the design of AI processors, Aldebaran (AB). He has special interests in many-core architecture, AI processor design, low-power processor design, fault tolerance design, and algorithmic optimizations of circuits and systems.

Minseok Choi received B.S. and M.S. degrees in electrical and electronics engineering from the Korea Advanced Institute of Science and Technology (KAIST), Republic of Korea in 1997 and 1999 respectively. He joined the Electronics and Telecommunications Research Institute (ETRI), Republic of Korea in 1999 and is currently a principle member of the research staff. His special interests include on-device deep learning processor hardware and software development.

Youngsu Kwon received B.S., M.S., and Ph.D. degrees from the Korea Advanced Institute of Science and Technology (KAIST), Republic of Korea in 1997, 1999, and 2004, respectively. He was a Postdoctoral Associate at the Microsystems Technology Laboratory (MTL), Massachusetts Institute of Technology from 2004 to 2005 designing 3-Dimensional FPGA. He has been with the AI SoC Research Department, Electronics and Telecommunications Research Institute (ETRI), Republic of Korea, since 2005. In ETRI, he is a Director, Principal Research Staff of the AI SoC Research Department devoted to the design of AI processors, AB. His special interests include many-core architecture, AI processor design, low-power architecture design, computer-aided design, and algorithmic optimizations of circuits and systems. He received the Presidential Prize from the Korean Government in 2016, Official Commendations from the Ministry of Science and ICT, as well as the Ministry of Industry in 2016, the Excellent Researcher Award from the Korea Research Council in 2013, the Industrial Contributor Award from the Korean Federation of SMEs in 2013, and medals from Samsung's Thesis Prizes in 1997 and 1999.