A 16 nm FinFET Heterogeneous Nona-Core SoC Supporting ISO26262 ASIL B Standard

Shinichi Shibahara, Chikafumi Takahashi, Kazuki Fukuoka, *Member, IEEE*, Yuko Kitaji, Takahiro Irita, Hirotaka Hara, Yasuhisa Shimazaki, *Member, IEEE*, and Jun Matsushima

***Abstract*— It is getting mandatory to comply with ISO26262 in the recent automotive development. The implementation of safety mechanism to detect faults is one of the keys in ISO26262. The fault prediction will make the automotive system more reliable. The SoC in this paper introduces two features: hardware built-in self-test (BIST) for safety mechanism supporting automotive safety integrity level (ASIL) B and killer-droop monitor for fault prediction. In addition, time slicing is introduced to the testing with hardware BIST during runtime so that each test session can be shorter than the required interrupt response time in the application. The killer-droop monitor can predict the voltage droop and stop the clock supply for a certain period of time to mitigate the droop for preventing delay faults on the SoC. The monitor samples the voltage based on time-to-digital converter at CPU operation clock and predicts the voltage from the history of sampled voltage. With that prediction feature, the minimum operation voltage can be improved by 50 mV at 2.02 GHz CPU operation, and the maximum frequency can be improved by 140 MHz under 0.82 V power supply in 16 nm process.**

***Index Terms*— BIST, fault prediction, functional safety, safety mechanism, time slicing, voltage droop, voltage prediction.**

I. Introduction

THE recent cars are supporting driver-assistance features, such as instrument cluster and surround view, which require high computational power. Car manufacturers are implementing safety features based on ISO26262 so that cars can detect faults inside and bring to safe state while driving [1]. Latest SoCs support automotive safety integrity level (ASIL) B by introducing BIST and redundant calculations [2].

This paper introduces an SoC for driver-assistance systems, consisting of nine heterogeneous CPUs, with supporting ISO26262 ASIL B standard. To reduce the hardware cost and satisfy the application requirements, hardware BIST is introduced to application processors for small area, and lock-step feature is introduced to real-time processor for a quick reaction to interrupt requests as safety mechanism. Hardware BIST is activated periodically so that faults can be detected during runtime, which is called RuntimeTEST.

In addition to the safety mechanism for faults, a fault prediction is introduced. Voltage droop can be one of the factors which cause faults. It is getting difficult to avoid the droop with design margins under high clock operation and low voltage supply. Killer-droop monitor can predict that voltage droop in advance and prevent faults by sending clock stop requests to the clock controller.

Fig. 1 shows a block diagram of the SoC which includes two clusters of quad-core application CPUs (Cortex-A53 and Cortex-A57), one real-time CPU (Cortex-R7), and one GPU (GX6650). Hardware BIST is attached to each module to check faults inside. Killer-droop monitor is embedded in each CPU of Cortex-A57 cluster. Fig. 2 shows the die photo and detailed specifications. This SoC is implemented with TSMC 16 nm FinFET process. The area is 111.36 mm^2 (12.94 mm $\times$ 8.61 mm). Cortex-A53, Cortex-A57, Cortex-R7, and GX6650 are operated at 1.2 GHz, 2 GHz, 800 MHz, and 700 MHz, respectively.

Section II visits the design for functional safety. Section III explains the mechanism of RuntimeTEST. Section IV details the killer-droop monitor. Section V shows the experimental results of applying RuntimeTEST and killer-droop monitor. Section VI summarizes this paper.

II. Design for Functional Safety

Functional safety is an approach to reduce the risk of harm to people such as injury. ISO26262 is an international standard for functional safety in road vehicle electrical and/or electronic systems. It describes the requirements for each ASIL, development flow, and its assessment. ASIL is classified from ASIL A to ASIL D, based on the safety requirements. To comply with such ASIL standard, introducing safety mechanisms is mandatory. Safety mechanisms are technical solutions to detect and react faults in order to maintain a safe state, according to ISO26262. Fault is also defined as abnormal condition that can cause failure. Fig. 3 shows the idea of fault reaction described in Part 1 of ISO26262. Safety mechanism detects faults such as random hardware fault. After that, the hardware status is returned to safe state. Detecting faults and returning to a safe state within the fault tolerant time interval (FTTI) window (window of time in which failures are caused by faults) are required in safety mechanism.

ISO26262 defines the criteria of diagnostic coverage for each ASIL standard. Guaranteeing the diagnostic coverage with safety mechanisms is mandatory. In ASIL B standard, 60% for latent fault and 90% for single point fault are required. Latent fault is a fault not detected by safety mechanisms. Single point fault is a fault which happens in the system. For example, error correction code (ECC) is one of the

Manuscript received May 1, 2016; revised September 30, 2016 and October 20, 2016; accepted October 24, 2016. Date of publication December 13, 2016; date of current version January 4, 2017. This paper was approved by Guest Editor Dejan Markovic.

S. Shibahara, C. Takahashi, K. Fukuoka, Y. Kitaji, and J. Matsushima are with Renesas System Design Co., Ltd., Tokyo 187-8588, Japan (e-mail: shinichi.shibahara.cj@renesas.com).

T. Irita, H. Hara, and Y. Shimazaki are with Renesas Electronics Corporation, Tokyo 187-8588, Japan.

Digital Object Identifier 10.1109/JSSC.2016.2623682

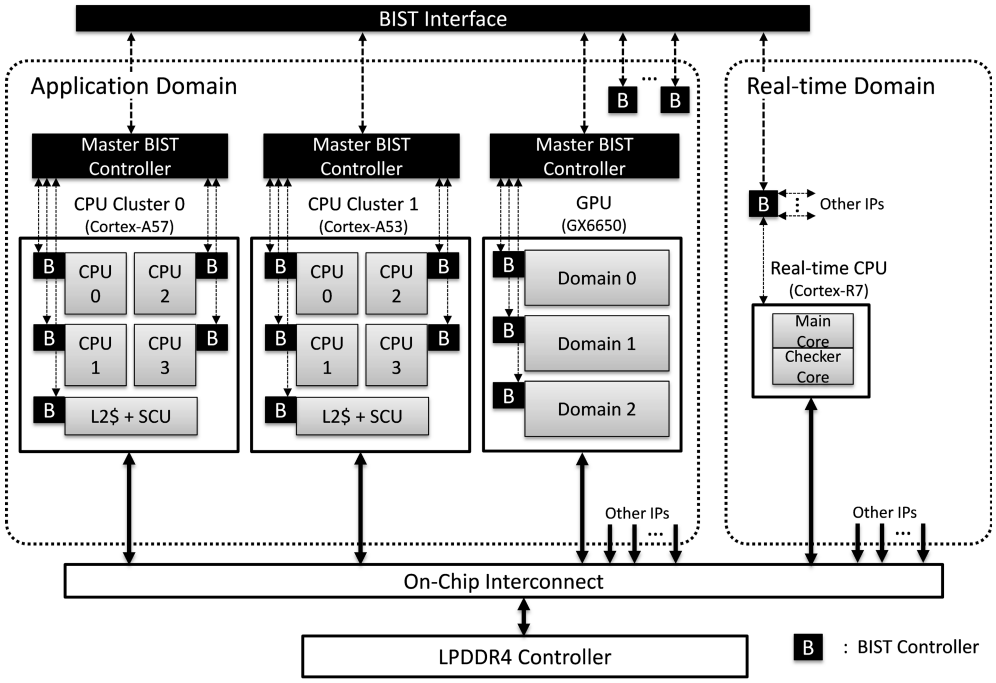
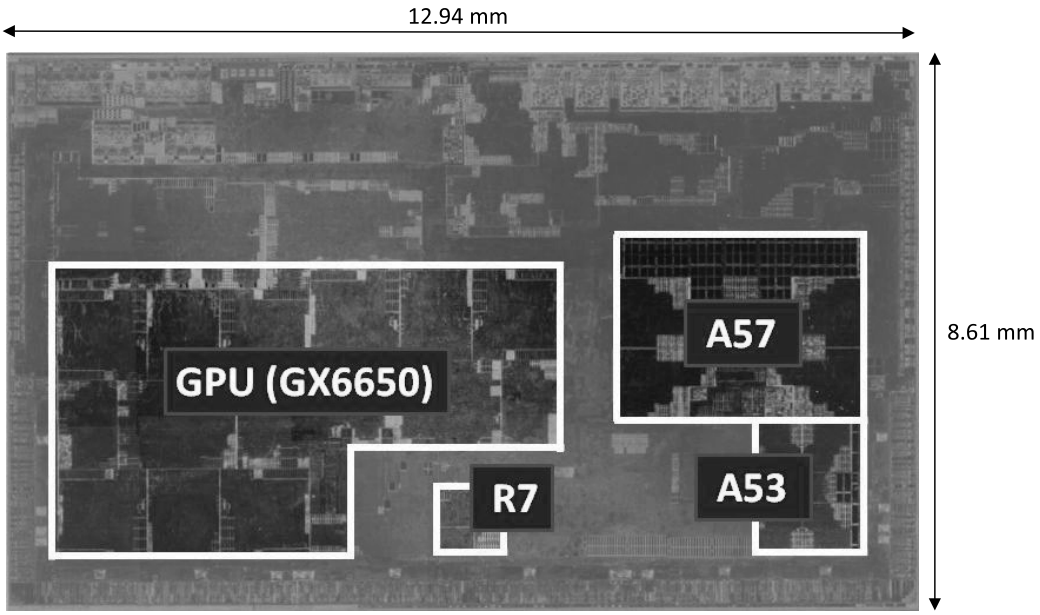


Fig. 1. Block diagram of the SoC.



Process	TSMC 16nm FinFET
Size	12.94mm x 8.61mm, 111.36mm^2
Memory	LPDDR4-3200

CPU	Quad Cortex-A57 2.0GHz
	Quad Cortex-A53 1.2GHz
	Dual-core Lock-step Cortex-R7 800MHz
GPU	PowerVR GX6650 700MHz

Fig. 2. Die photo and specifications.

safety mechanisms for embedded memories. It can detect at most two-bit errors at memory read operations. In addition, hardware BIST is introduced to detect faults in the ECC encode and decode logics as latent fault detection mechanism.

Typically, redundant hardware, self-test by software, and self-test supported by hardware are introduced as safety mechanism. Redundant hardware is the most reliable method, because it can detect faults by monitoring different behaviors between the target and its duplicated hardware at the expense of extra logic. Software self-test does not need the extra logic, but it requires long time, which causes an interruption of system function during runtime. Hardware-based

TABLE I
COMPARISON IN FAULT DETECTION COVERAGE, AREA, AND INTERRUPT RESPONSE

	No measure	Dual Core Lock Step (Redundant HW)	Software Self Test	Hardware Supported Self-Test	This work
Fault detection coverage	N/A	High	Low	Medium	Medium
Relative logic size of CPU cluster	1.00	> 2.30	1.00	< 1.10	< 1.10
Blackout time for interrupt response	N/A	N/A	> 100ms	> 10ms	< 2.0ms

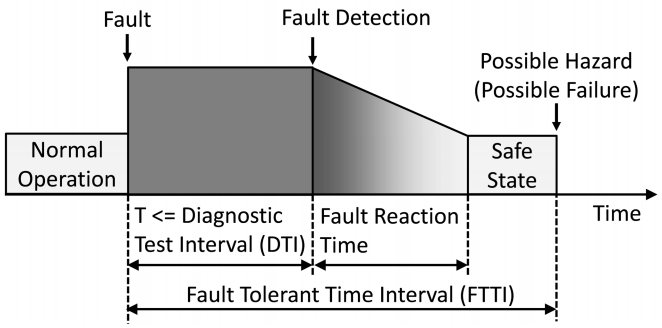


Fig. 3. Fault reaction after fault detection [1].

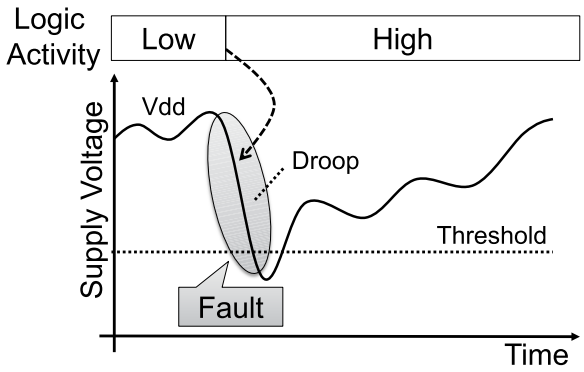


Fig. 4. Fault from voltage droop.

self-test can reduce test time, but it increases area on chip.

The SoC in this paper is for driver-assistance systems which are required to support ASIL B standard. There are many choices to support ASIL B standard in terms of implementation while most systems complying with ASIL D standard introduce duplication in hardware or software. Generally, the SoC with high computational power consumes a lot of area. This paper describes the best choice to meet the system requirements, such as performance and interrupt response time with limiting the area overhead in safety mechanisms.

Several safety mechanisms are introduced to processors for application requirements. One CPU in real-time domain applies dual-core lock-step. The dual-core lock-step consists of two identical CPUs so that wrong operations can be identified by comparing the output of two CPUs. The benefit is that the CPU does not need to stop the operation for detecting faults during runtime. The real-time CPU requires the early interrupt response and quick interrupt handling. The dual-core lock-step implementation satisfies those requirements. The CPUs in application domain apply parity and ECC for memories, and hardware BIST for logics. One GPU applies hardware BIST. Parity and ECC implementation can detect faults at memory read operations. Hardware BIST is activated while the target CPU is not operated. Introducing hardware BIST can save the logic cost compared with lock-step implementation.

The test time for fault detection is a key for functional safety. The diagnostic test interval (DTI) in Fig. 3 is defined based on the application requirements. The scan testing is an efficient way to detect faults with a small pattern set. The recent electronic design automation (EDA) tools can generate the scan patterns, which can reduce the human effort, compared with preparing software self-test. In addition, in CPUs with hardware BIST, the interrupt response time is also an important factor. For example, in audio playback, the interrupt response time must be less than 2 ms. To accommodate such requirement, it is desirable that the test session with hardware BIST can be split into small sessions. The SoC in this paper supports the RuntimeTEST feature with time slicing function. The details are explained in Section III.

Fault reaction after detecting faults needs additional time for the recovery to safe state. The application running on the SoC cannot be operated during the recovery. It is desirable to predict faults and react for the coming faults in advance, which can avoid the nonoperational situation. Voltage droop is one of the factors which causes delay faults. Fig. 4 shows the mechanism. Voltage droop happens when the logic activity in the circuits gets high. The voltage droop can be avoided by introducing the timing margin to the design, but it is getting difficult in the recent process technology because of the complexity. The 16 nm process introduces FinFET transistors instead of conventional bulk transistors to achieve high-frequency operation under low supply voltage. However, the dynamic current cannot be reduced in the higher performance design with less timing margin, which induces larger dynamic current and higher current density. As a result, the voltage droop coming from large current induction needs to be cared. The SoC in this paper introduces a killer-droop monitor. The killer-droop is a critical voltage drop causing a delay fault. The killer-droop monitor can predict the killer-droop in advance, and send a request to stop the clock supply until the droop is mitigated. It can avoid the fault with additional performance cost in the application. The details are discussed in Section IV.

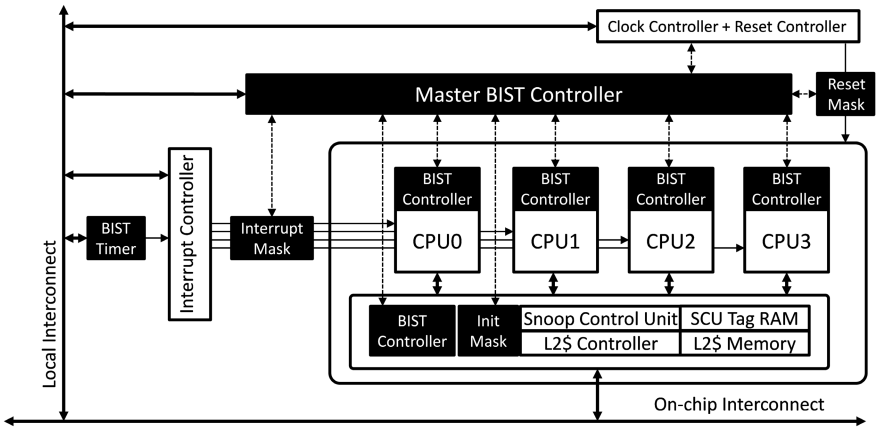


Fig. 5. Block diagram of RuntimeTEST implementation.

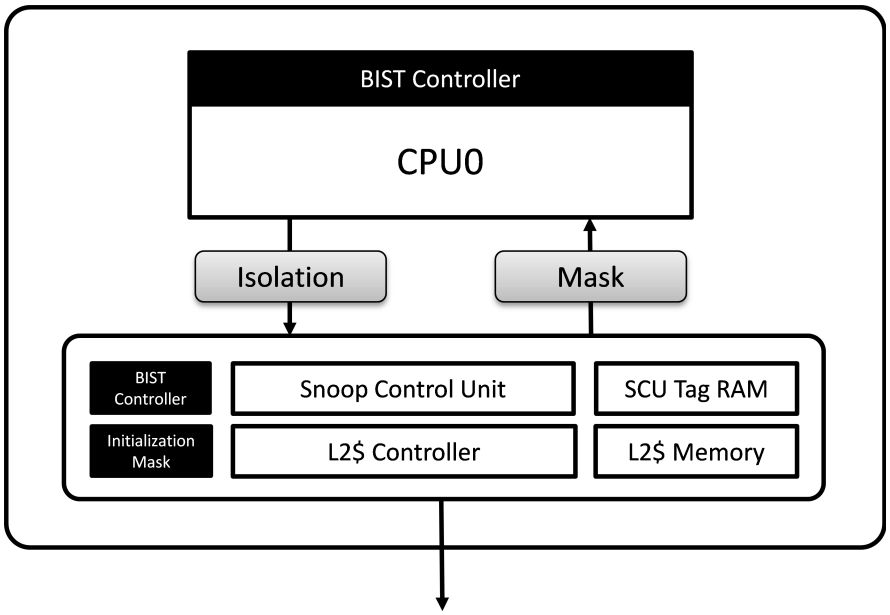


Fig. 6. Power isolation and input mask for CPU RuntimeTEST.

III. RuntimeTEST

Hardware BIST is utilized as safety mechanism for detecting faults. At system power-on, hardware BIST works for power-on self-test to detect latent fault. During runtime, hardware BIST is the key element of RuntimeTEST to detect single point fault. Hardware BIST can detect fault such as stack-at fault from transistor aging. Introducing hardware BIST can reduce the area compared with lock-step implementation, and avoid the effort to create software self-test. Hardware BIST is applicable when the duplication of the target module exceeds the budget in chip area and power, and the target module is not always required during runtime.

After running RuntimeTEST, all elements in the test target need to be initialized, because hardware BIST manipulates storage elements such as flip-flops during scan test. It will become a performance penalty because the restoration is required to resume the application. Multicore processor supporting hardware cache coherency includes duplicated tag memories for level-1 data cache on each CPU. If those tag memories are initialized after running RuntimeTEST, all CPUs need to be initialized to keep the cache coherency. Generally, multilevel cache is introduced to reduce the cache miss latency on level-1 cache in CPU. As the level of cache hierarchy is lower, the cache size is bigger. If the lower level cache is initialized, the cache miss latency on level-1 cache becomes

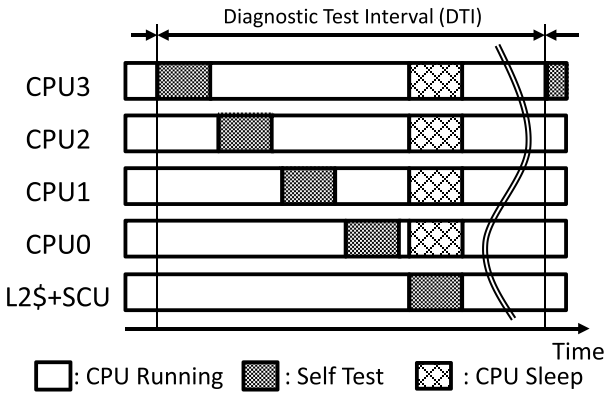


Fig. 7. RuntimeTEST timing chart.

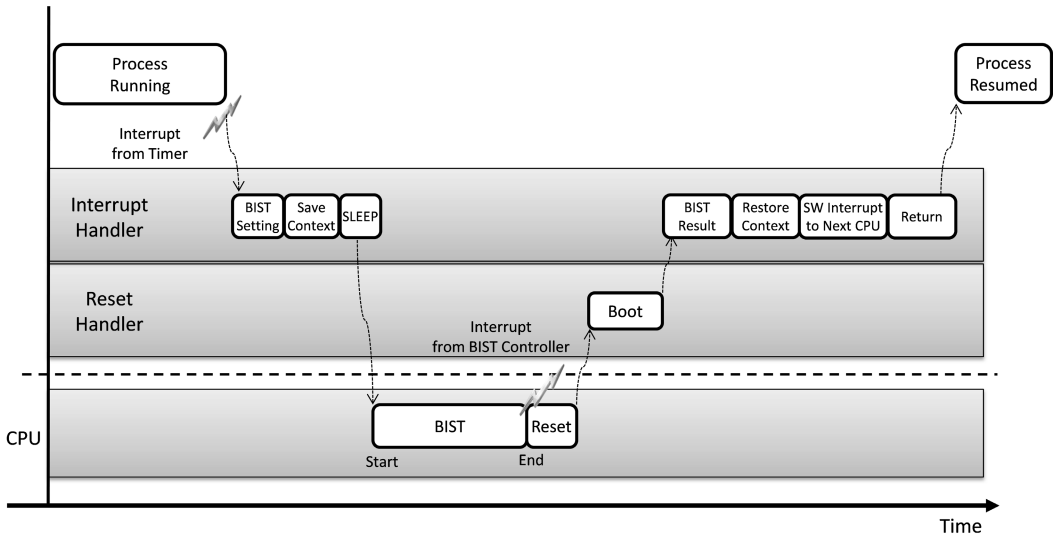


Fig. 8. RuntimeTEST procedure for CPU.

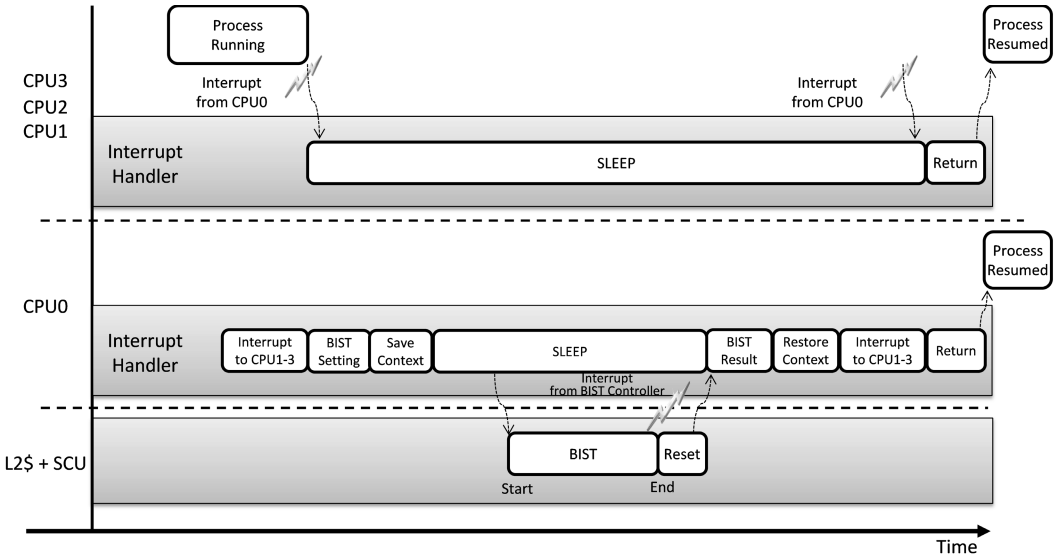


Fig. 9. RuntimeTEST procedure for common block.

larger. Because of such nature, large scale SoCs, including multicore and multilevel cache, require the way to minimize the performance penalty.

The logic scale of CPU and GPU is getting bigger to get the higher performance. The test time with hardware BIST is also getting longer. However, the application running on processor requires that the execution time of RuntimeTEST must be less than the required interrupt response time. For example, real-time OS requires less than 500 μs as interrupt response time. Audio playback needs less than 2 ms as interrupt response time to play continuously. In SoCs requiring high performance, because of the large logic scale, the test time is usually longer than the required execution time. To meet such application requirements, it is desirable that SoCs have the way to limit the test time.

The SoC in this paper introduces two ideas to the RuntimeTEST feature. The first idea is that the test target is separated based on power domain in the SoC. Each power domain is tested exclusively while all domains are tested in parallel during power-on self-test. In the case of multicore processor, each CPU and the common block, such as level-2 cache and snoop control unit (SCU), are not tested in parallel. Even if one CPU is in RuntimeTEST, the other CPUs are still able to perform.

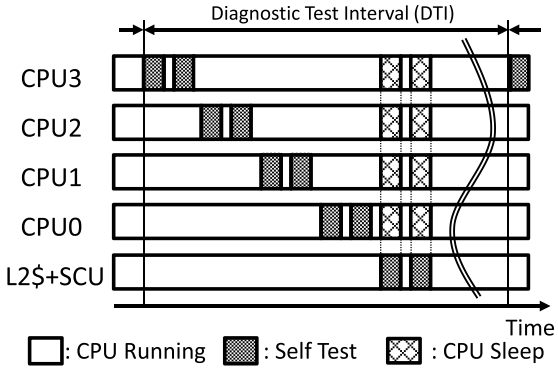


Fig. 10. RuntimeTEST timing chart with time slicing.

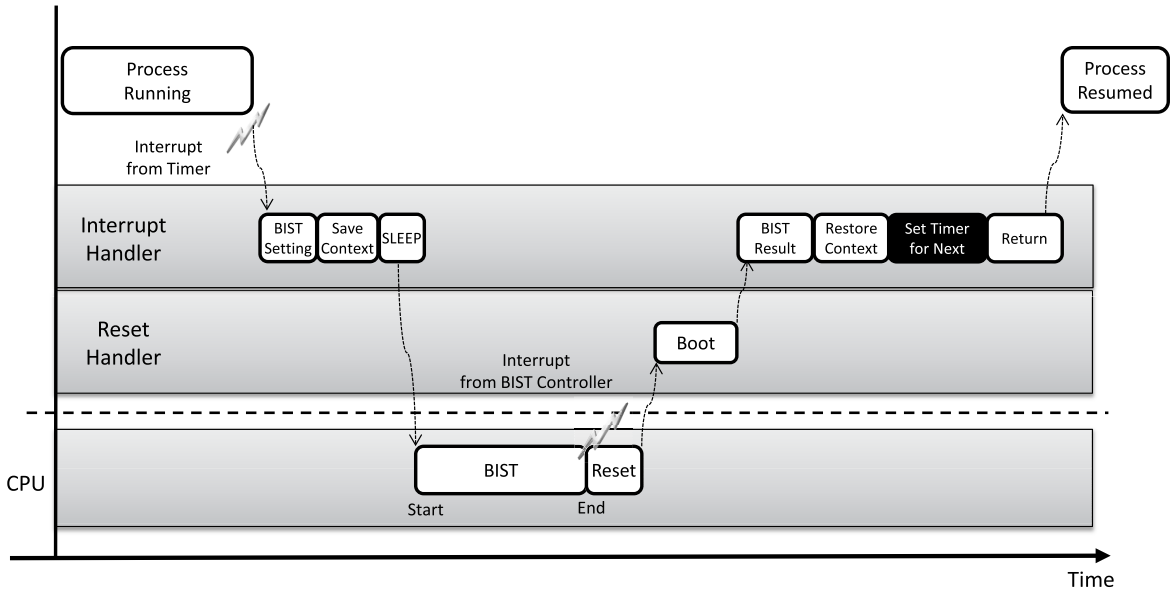


Fig. 11. RuntimeTEST procedure for CPU with time slicing.

It means that tasks running on the CPU can be migrated to the other CPUs if required. Also, during the RuntimeTEST in the common block, the memory contents for level-2 cache and snoop control unit are preserved so that all CPUs can resume own tasks with no penalty after the RuntimeTEST, and the cache miss penalty on level-1 cache can be reduced. With utilizing the nature of multicore and multilevel cache, the performance penalty from RuntimeTEST is minimized. The second idea is that the RuntimeTEST session can be split into sessions with time slicing. The hardware BIST controller has the control register for setting the duration of test time. With adjusting the control register, the requirement in interrupt response time can be satisfied.

Table I shows the comparison of fault detection coverage, area, and interrupt response among several implementations for safety mechanism [3], [4]. The SoC in this paper can control the fault detection coverage and test time based on ASIL standard and application requirements with small area cost.

A. RuntimeTEST Implementation

Fig. 5 shows the block diagram of RuntimeTEST implementation for multicore processor. A BIST controller is attached to each power domain in the processor such as CPU core and common block consisting of level-2 cache and snoop control unit. The BIST controller has three functions: logic BIST, memory BIST, and conventional scan test. Logic BIST and memory BIST are applied to the power-on self-test for detecting latent fault. In RuntimeTEST, only logic BIST is applied, because the faults in memories can be detected at memory read operations because of memory protection with ECC and parity. Conventional scan test is used when fault detection coverage needs to be achieved in shorter time than logic BIST in RuntimeTEST. Scan test patterns generated from logic BIST are based on random. Conventional scan test with directed scan test patterns is easier to reach to the target fault

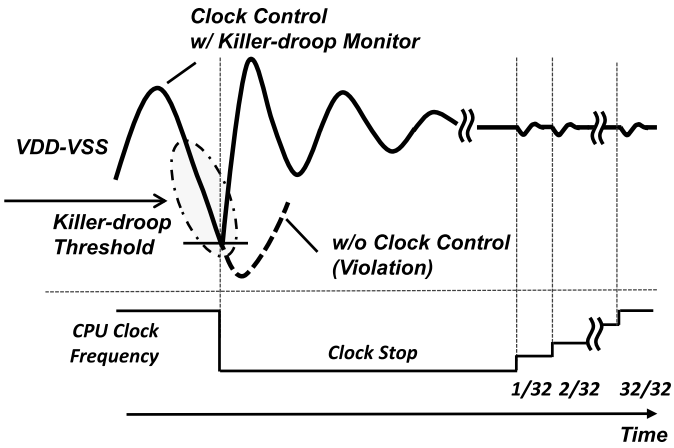


Fig. 12. Fault avoidance with killer-droop monitor.

detection coverage. The master BIST controller controls all BIST controllers. It has a scan control register to set the test type (logic BIST or memory BIST or conventional scan test) and the test time. The master BIST controller sends commands to BIST controllers and receives the test result from BIST controllers. In the case of conventional scan test, master BIST controller fetches the scan test patterns from memories, such as internal SRAMs, and sends to BIST controllers. The BIST timer sends an interrupt request to the interrupt controller every DTI based on application requirements for supporting ISO26262 as shown in Fig. 3. The interrupt mask is to mask interrupt requests from peripheral modules to the CPU under RuntimeTEST. The reset mask is to mask the reset request to CPUs after RuntimeTEST for the common block. The RuntimeTEST implementation on the SoC utilizes the implementation for power management. When the common block is in RuntimeTEST, the power management modules, such as clock controller and reset controller, recognize that the block is in power off state although the block is still powered on for RuntimeTEST. Because of that implementation, the reset controller sends the reset request to all CPUs as

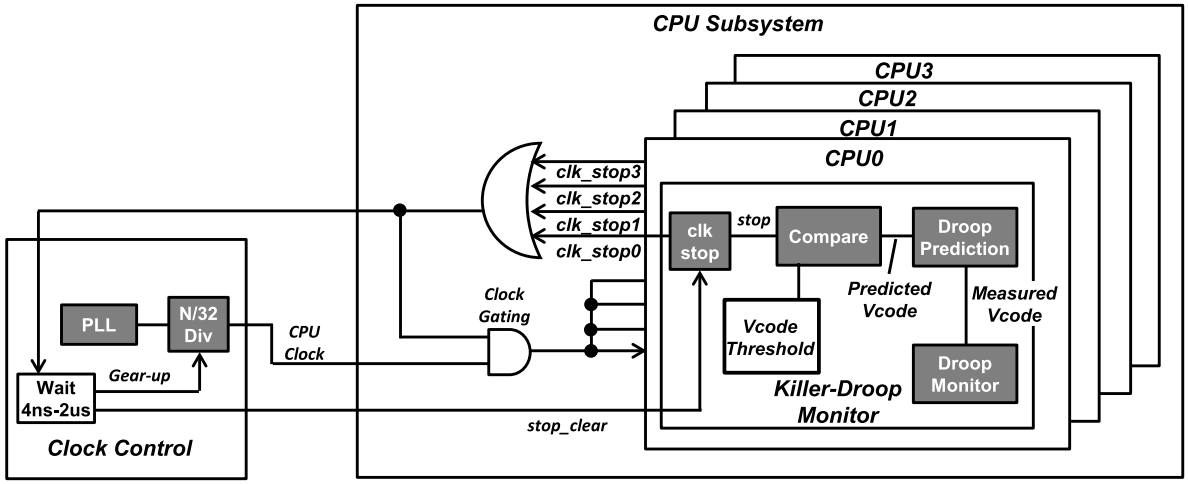


Fig. 13. Adaptive clock control with killer-droop monitor.

well as common block after RuntimeTEST. The reset mask is to avoid that reset request to resume processes on CPUs. The initialization mask is to block the initialization to level-2 cache memories and SCU tag RAMs after RuntimeTEST for the common block. The processors in the SoC have the feature to initialize cache memories after the reset. The initialization mask is to avoid that initialization to preserve the memory contents.

Fig. 6 shows the additional implementation around each power domain for RuntimeTEST. The power isolation cells for power domain separation can be utilized to mask the propagation of toggles from flip-flops to different power domains during scan test in logic BIST. In addition, each input port on each power domain has a mask cell, because the expectation values of scan patterns cannot be determined without fixing all input ports to specific values.

B. RuntimeTEST Procedure

The power domain under RuntimeTEST needs to be isolated from the other power domains. After RuntimeTEST, the power domain needs to be initialized. Those situations are similar to the power-down and power-on procedures. The RuntimeBIST utilizes the power management feature because of that similarity. With that utilization, the logic implementation for RuntimeTEST can be minimized. Also, the software for RuntimeTEST procedure can reuse from the software for power management. In symmetric multicore processor, programs running on CPUs can be migrated to the other CPUs. To minimize the time that no program is executed on CPUs, only one CPU executes RuntimeTEST at one time. Meanwhile, the tasks on the CPU can be migrated to the other CPUs.

Fig. 7 shows the timing chart of RuntimeTEST on one CPU cluster. Each CPU executes RuntimeTEST in a different time. When the common block executes RuntimeTEST, all CPUs need to stop the execution, because all CPUs refer to the common block. The total time of execution time on all power domains needs to be less than DTI for ISO26262. Fig. 8 shows the procedure to start RuntimeTEST and exit from RuntimeTEST for CPU core. When the test target CPU receives the interrupt request from BIST timer and jumps to

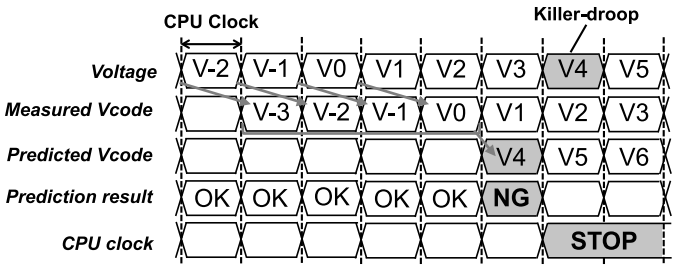


Fig. 14. Voltage prediction with voltage sampling.

the interrupt handler, the test target CPU starts with setting the control registers on the master BIST controller. The remaining procedure is the same as power down sequence in low-power management. After the target CPU goes to sleep state for RuntimeTEST, the master BIST controller detects the sleep state and invokes the BIST controller attached to the CPU to start the hardware BIST. Meanwhile, mask cells and isolation cells are activated as if the CPU is powered off. After finishing the hardware BIST, the BIST controller sends the result to the master BIST controller. The master BIST controller sends the interrupt request to the target CPU like the trigger of power on. The reset controller monitors that request and sends the reset request to the target CPU. The mask cells and isolation cells are deactivated as if the CPU is powered on. The initialized CPU starts with reset handler and goes back to the interrupt handler. In the interrupt handler, the CPU confirms the test result by referring to the registers in the master BIST controller. When the hardware BIST is successful, the CPU restores the context which is saved before starting the hardware BIST, sends the software interrupt to the next target CPU, and returns to the process previously executed. If the hardware BIST is not successful, the fail result is notified to the other CPUs for fault reaction, such as reinitializing or disabling the defect CPU. After receiving the interrupt request at the next target CPU, the procedure for RuntimeTEST is the same as the previous target CPU. The procedure for exiting from hardware BIST can reuse the code for power-on sequence because of the similarity.

Fig. 9 shows the procedure to start RuntimeTEST for the common block. After the last target CPU finished the hardware

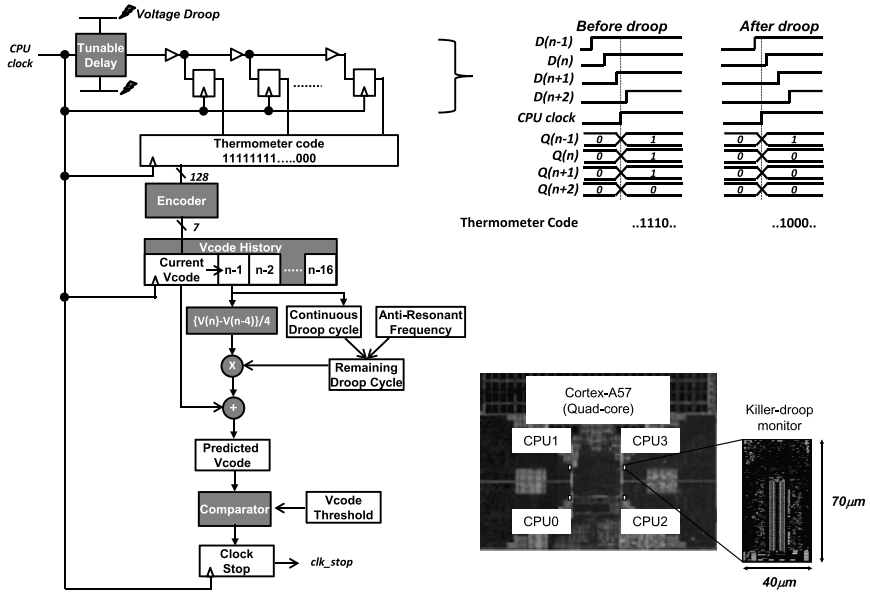


Fig. 15. Killer-droop monitor implementation.

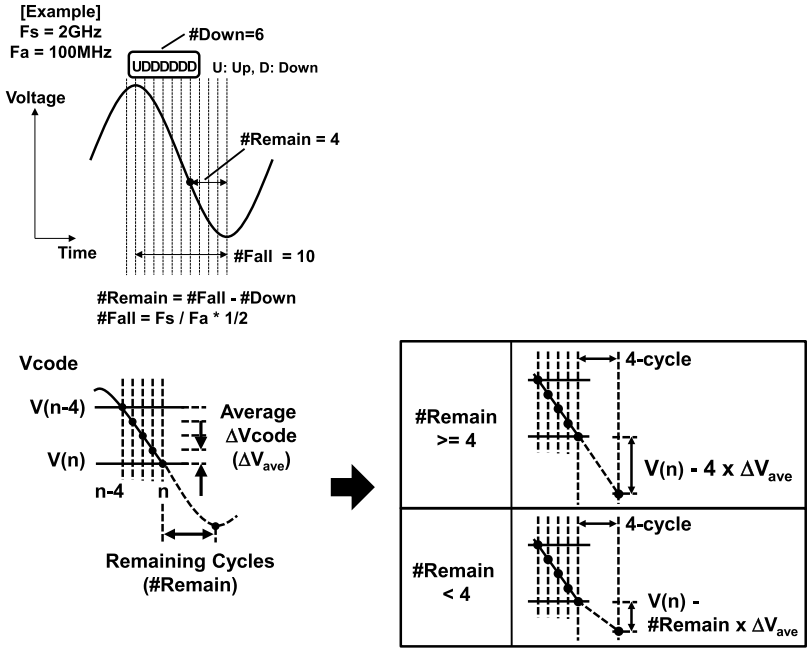


Fig. 16. Voltage prediction in killer-droop monitor.

BIST, the CPU sends a software interrupt request to the other CPUs. The other CPUs receive the interrupt request and go to sleep state. Meanwhile, the last CPU starts the hardware BIST settings for the common block. The hardware BIST for the common block is started when the last CPU saves the necessary context for the common block and goes to sleep state. The behavior of mask cells and isolation cells in the common block is the same as those cells in CPU. The reset controller sends the reset request to the common block after the hardware BIST is finished. And, the master BIST controller sends the interrupt request to the last CPU. The last CPU takes care of confirming the BIST result. Also, the last CPU sends the interrupt request to the other CPUs so that all CPUs can resume own processes executed before.

C. RuntimeTEST Procedure With Time Slicing

To adjust the test time for the interrupt response time requirement, the hardware BIST session can be split with time slicing. Figs. 10 and 11 show the timing chart and the procedure. With introducing the time slicing, the session can be split into multiple sessions. To terminate and resume the remaining session later, instead of sending the interrupt request to the next target CPU before exiting from RuntimeTEST as in Fig. 8, another BIST timer not in Fig. 5 is set as the next trigger to the remaining session. One drawback in the time slicing method is that the execution time for start and exit procedure needs to be added to each session, which gets longer in the total test time for RuntimeTEST. The tradeoff between

TABLE II
COMPARISON IN DROOP MONITORING AND REACTION

	This work	ISSCC2014[5]	ISSCC2014[6]	ISSCC2015[7]
Technology	16nm	28nm	28nm	16nm
Application	Automotive	Mobile	PC	Mobile
Droop Monitor	TDC (Time to Digital Converter)	Ring oscillator	DLL	Replica path delay
Monitor Interval	1-cycle	50ns	1-cycle	1-cycle
Reaction Latency (Start Reaction after Droop)	0-cycle	100ns	2-cycle	1.5-cycle
Reaction Mechanism	Clock stop	Frequency down	Clock pulse stretch	Frequency down
Droop Prediction	Yes	n/a	n/a	n/a

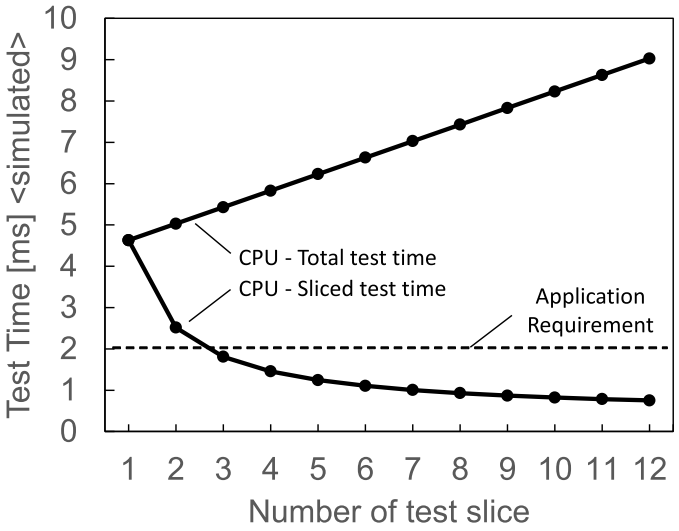


Fig. 17. Test time in RuntimeTEST for Cortex-A57 CPU.

single test time with time slicing and total test time needs to be considered during the system development.

IV. KILLER-DROOP MONITOR

The voltage droop can be one of the reasons for wrong operations. The killer-droop is a critical voltage drop causing a delay fault. It is getting difficult to avoid the voltage droop because of the low power and high performance design which requires to reduce the design margin. The fault prediction for the fault avoidance will make the system more reliable. The killer-droop monitor in this paper can predict the killer-droop in advance and take measures for avoiding the faults. The combination of fault prediction and prevention is more effective than fault detection, because it can continue the current task without retries.

Fig. 12 summarizes the idea of fault avoidance with killer-droop monitor. When the monitor predicts that the voltage goes below the threshold, the monitor sends a request to stop the clock supply to avoid the fault operation until the droop becomes acceptingly small. Then, the clock supply is resumed with 1/32 the frequency of the original clock, and the clock frequency is increased gradually to avoid the further voltage droop. Fig. 13 shows the detailed implementation of adaptive clock control in the killer-droop monitor. The killer-droop monitor is embedded in each CPU. Once the killer-droop monitor predicts the killer-droop, the monitor sends the clock stop request to the clock controller. After the clock controller stops the clock supply for a certain period of time, the controller resumes the clock supply by controlling the clock frequency with clock divider. The threshold voltage and the duration for stopping the clock supply are determined by circuit-level simulation with introducing the factor of on-die capacitance and package inductance.

The killer-droop monitor has three key features compared with prior works as in Table II [5]–[7]: 1) voltage sampling at CPU operation frequency; 2) voltage-droop prediction; and 3) ability to stop and resume the clock supply from a low clock frequency. The monitor interval means the interval that the monitor predicts the voltage. The reaction latency is the latency that the monitor starts the reaction after the killer-droop. The killer-droop monitor starts to stop the clock supply before the killer-droop. The reaction latency can be practically 0-cycle because of prediction.

A. Killer-Droop Monitor Implementation

The killer-droop monitor predicts the voltage with sampling voltage at the CPU operation frequency. Fig. 14 summarizes the idea of voltage prediction. This monitor measures the potential difference between core voltage (V_{DD}) and ground (V_{SS}) in the CPUs and outputs it as V_{code} (coded differential voltage), which is used in the droop prediction logic.

Fig. 15 shows the voltage monitor based on time-to-digital converter (TDC) consisting of buffer chains and flip-flops. The tunable delay is to compensate the process variation. The voltage (V_{code}) can be measured by comparing a phase of monitor clock signal with the phase of a delay clock signal. The history buffer (V_{code} History) samples that voltage every cycle. Minimizing the parasitic *RC* influence in the TDC by noise isolation and equal-length wiring achieves 5 mV voltage resolution. The size of killer-droop monitor is 40 μm $\times$ 70 μm.

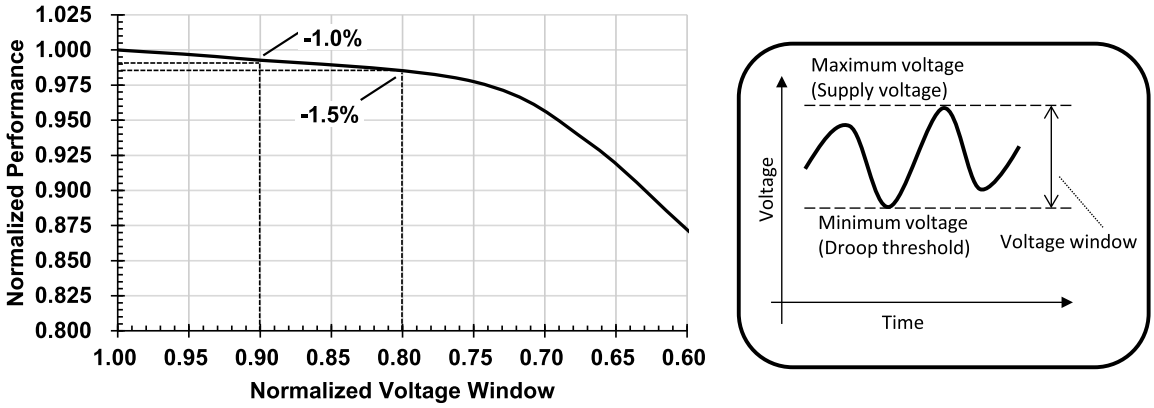


Fig. 18. Performance against voltage window.

Frequency [GHz]		1.78	1.82	1.85	1.88	1.92	1.95	1.98	2.02
Predicted	8-cycle	P	P	P	F	F	F	F	F
	4-cycle	P	P	P	P	P	P	P	F
	2-cycle	P	P	P	P	P	F	F	F
	1-cycle	P	P	P	F	F	F	F	F
Measured		P	P	P	F	F	F	F	F
w/o Clock Control		P	F	F	F	F	F	F	F

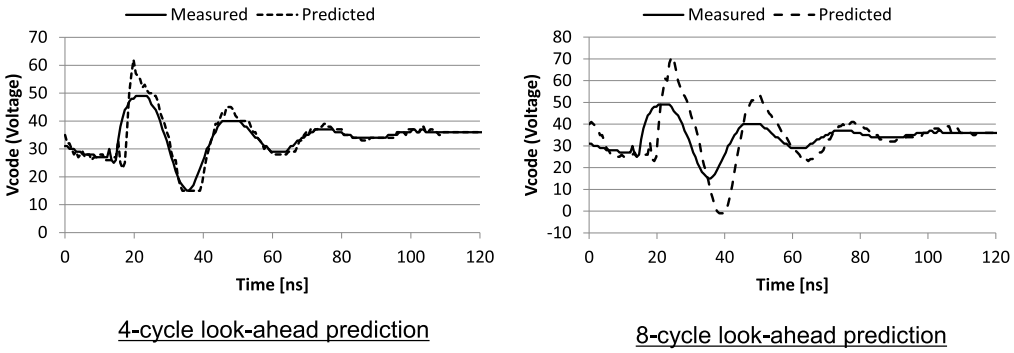


Fig. 19. Effect and accuracy of voltage prediction under 0.80 V supply.

B. Voltage Prediction

With the history of V_{code}, the voltage can be predicted every cycle. Fig. 16 summarizes the idea of voltage prediction in killer-droop monitor in the case of 4-cycle look-ahead prediction. The voltage transition over time is supposed to have a periodical characteristic with antiresonant frequency. The antiresonant frequency is calculated through circuit-level simulation considering on-die capacitance and package inductance. When the antiresonant frequency (F_a) is 100 MHz under 2 GHz sampling frequency (F_s), the chances to see that the voltage is going down (*#Fall*) are ten times. By recording the voltage transition, the monitor can predict how the voltage is changed in the future. When the voltage is going down (*#Down*) for six cycles, the monitor predicts that the voltage will continuously go down for additional four cycles (*#Remain*). The voltage is predicted based on *#Remain* and the average voltage reduction in cycle (ΔV_{ave}) with referring to the history of V_{code}. If *#Remain* is less than four, the further voltage reduction is *#Remain* times of ΔV_{ave}. Otherwise, the reduction is four times of ΔV_{ave}.

The killer-droop monitor cares for the worst case in voltage droop. The worst case happens when the voltage oscillates at the antiresonant frequency. The antiresonant frequency can be adjusted by on-die capacitance and package inductance so that the monitor can detect the killer-droop.

V. Experimental Results

To confirm the effect of two features in the SoC, several experiments are conducted.

A. RuntimeTEST

The total test time of RuntimeTEST for Cortex-A57 CPU with time slicing is shown in Fig. 17. The total test time is calculated by the summation of the test time for hardware BIST and the time for start and exit of RuntimeTEST. The total test time to reach to 90% fault coverage without

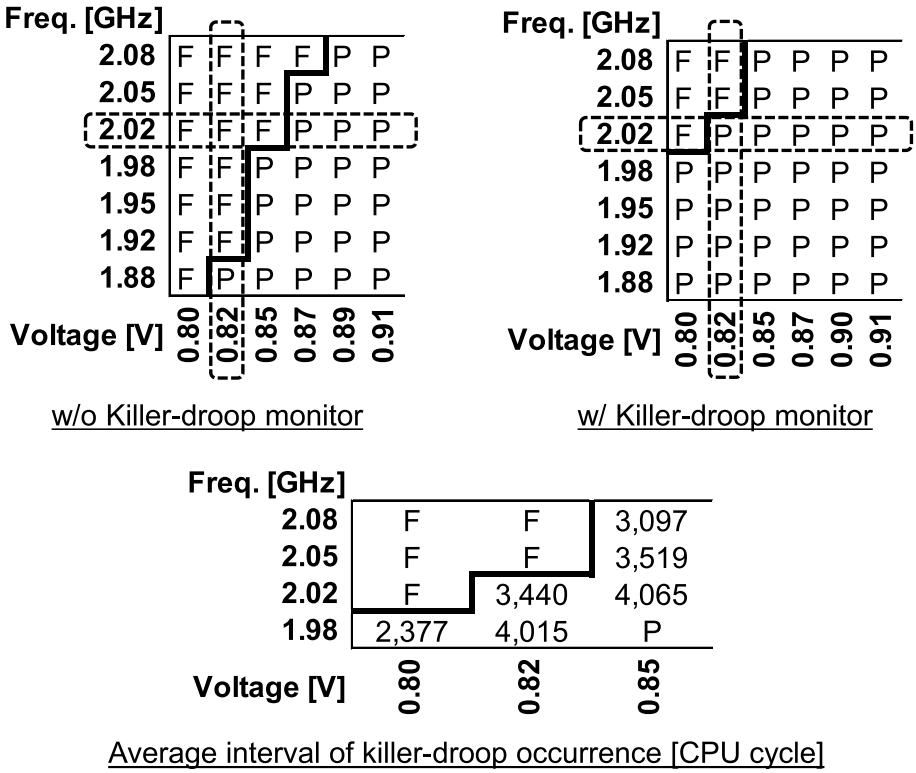


Fig. 20. Effect of killer-droop monitor.

time slicing is 4.63 ms, including 0.4 ms for start and exit of RuntimeTEST. If the required interrupt response time is less than 4.63 ms, the time slicing is necessary. As the number of time slice is increased, the test time for each session is decreased. However, the total test time is increased because of the execution time for start and exit of RuntimeTEST in each session. In the case of audio playback whose required interrupt response time is 2 ms, slicing into three sessions is the best choice.

B. Killer-Droop Monitor

The setting of minimum voltage threshold affects to the performance due to the duration of stopping the clock supply. The performance degradation against the minimum voltage threshold setting is shown in Fig. 18. The x-axis and the y-axis in the graph are the voltage window (difference between maximum and minimum voltage) and the CPU performance normalized by the measured result without clock stoppage. When the threshold V_{code} (killer-droop threshold) is high, the voltage variation can be minimized. However, the clock supply is stopped frequently, which causes the performance degradation. That said, even when the voltage window is reduced by 10% and 20%, the performance degradation is only 1% and 1.5%, respectively. This result demonstrates that killer-droop can be avoided while maintaining performance.

Fig. 19 confirms the effect and accuracy of voltage prediction with changing the prediction range. The accuracy of voltage prediction is measured under the condition that a program with high load is running on the Cortex-A57 cluster. Measured means the 0-cycle look-ahead prediction. The shmoo plot shows that the 4-cycle look-ahead prediction brings the best performance. The 8-cycle prediction is pessimistic, which causes the performance degradation. The error in 4-cycle look-ahead prediction is quite small except the situation that the voltage is high. The error in the high voltage is not a problem, because the killer-droop monitor stops the clock supply only when the voltage is low.

Fig. 20 shows the performance improvement and supply voltage reduction with killer-droop monitor in shmoo plot. With killer-droop monitor and its fault avoidance reaction with measured interval, 50 mV voltage reduction can be achieved under the same operation frequency condition. Also, the operation frequency can be improved by 140 MHz under the same voltage condition.

VI. Conclusion

This paper presents two features for supporting functional safety in the 16 nm SoC: RuntimeTEST and killer-droop monitor. RuntimeTEST is introduced for fault detection to support ISO26262 ASIL B standard. To meet the application requirement in the interrupt response time, time slicing is introduced to RuntimeTEST. Killer-droop monitor is introduced for fault prediction in voltage droop. With predicting the voltage, the killer-droop can be prevented. The experimental result demonstrates that RuntimeTEST can be introduced to application processors with satisfying the application requirements. Also, introducing killer-droop monitor can reduce the supply voltage and improve the operation frequency.

REFERENCES

[1] "Road vehicles—Functional safety," ISO 26262, International Organization for Standardization, 2011.

[2] Z. Nikolic *et al.*, "A scalable heterogeneous multicore architecture for ADAS," in *Proc. Hot Chips*, Aug. 2015.

[3] H. Al-Asaad, B. T. Murray, and J. P. Hayes, "Online BIST for embedded systems," *IEEE Design Test Comput.*, vol. 15, no. 4, pp. 17–24, Oct. 1998.

[4] L. Chen and S. Dey, "Software-based self-testing methodology for processor cores," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 20, no. 3, pp. 369–380, Mar. 2001.

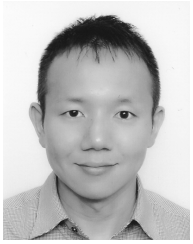
[5] M. Igarashi *et al.*, "A 28nm HPM heterogeneous multi-core mobile application processor with 2GHz cores and low-power 1GHz cores," in *ISSCC Dig. Tech. Papers*, Feb. 2014, pp. 178–179.

[6] A. Grenat, S. Pant, R. Rachala, and S. Naffziger, "Adaptive clocking system for improved power efficiency in a 28nm x86-64 microprocessor," in *ISSCC Dig. Tech. Papers*, Feb. 2014, pp. 106–107.

[7] K. Bowman *et al.*, "A 16nm auto-calibrating dynamically adaptive clock distribution for maximizing supply-voltage-droop tolerance across a wide operating range," in *ISSCC Dig. Tech. Papers*, Feb. 2015, pp. 152–153.

Shinichi Shibahara received the M.E. degree in informatics and mathematical science from Osaka University, Osaka, Japan, in 2000.

He joined Hitachi, Ltd., Tokyo, Japan, in 2000. He was with Renesas Technology Corporation, Tokyo, and Renesas Electronics Corporation in 2003 and 2010, respectively. He was a Visiting Researcher with the University of California at Irvine, Irvine, CA, USA, from 2007 to 2008. Since 2015, he has been with Renesas System Design Co., Ltd., Tokyo. He has been involved in the design and development of CPUs and CPU subsystems from low-end general purpose MCUs to high-end mobile SoCs. He is currently leading the development of CPU subsystems for automotive SoCs.



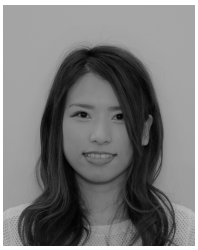
Chikafumi Takahashi received the B.E., M.E., and Ph.D. degrees in engineering from the University of Tsukuba, Tsukuba, Japan, in 2002, 2004, and 2008, respectively.

He joined Renesas Technology Corporation, Tokyo, Japan, in 2008. Since 2015, he has been with Renesas System Design Co., Ltd., Tokyo, where he is involved in the development of CPU subsystems for automotive SoCs and Internet of Things-oriented microcontrollers.

Kazuki Fukuoka (M'07) received the B.E., M.E., and Ph.D. degrees in electrical and electronic engineering from Kobe University, Kobe, Japan, in 1998, 2002, and 2005, respectively.

He was with Sharp Corporation, Osaka, Japan, where he was involved in RF circuits design from 1998 to 1999. He joined Renesas Technology Corporation, Tokyo, Japan, in 2005. He was with Renesas Electronics Corporation, Tokyo, in 2010. Since 2015, he has been with Renesas System Design Co., Ltd., Tokyo. He has been involved in developing power management techniques from 90- to 16-nm nodes. Currently, he is responsible for low-power and high-speed CMOS circuits and power delivery networks in advanced technologies.

Dr. Fukuoka has been a member of the IEEE SSCS since 2007.



Yuko Kitaji received the B.E. and M.E. degrees in electronic and photonic engineering from the Kochi University of Technology, Kochi, Japan, in 2007 and 2009, respectively.

She joined NEC Electronics Company, Ltd., Kawasaki, Japan, in 2009. She was with Renesas Electronics Corporation, Tokyo, Japan, in 2010. Since 2015, she has been with Renesas System Design Co., Ltd., Tokyo. She has been involved in low-power CMOS digital circuit design.

Takahiro Irita received the B.E. and M.E. degrees in electronic engineering and the D.Eng degree in information and communication engineering from the University of Tokyo, Tokyo, Japan, in 1993, 1995, and 1998, respectively.

He joined Hitachi, Ltd., Tokyo, in 1998. He was with Renesas Technology Corporation, Tokyo, in 2003. He is currently with Renesas Electronics Corporation, Tokyo. His current research interests include the SoC architecture for car information system.

Hirotaka Hara received the B.E. degree in pure and applied science from the University of Tokyo, Tokyo, Japan, in 1987, and the M.S. degree in electrical engineering from the California Institute of Technology, Pasadena, CA, USA, in 1997.

He was with Hitachi, Ltd., Tokyo, in 1987, where he was involved in SoC designs and developments. Since 1999, he has been involved in the automotive information SoC projects. He is currently a Senior Chief Professional of the first Solution Business Unit, Renesas Electronics Corporation, Tokyo.

Yasuhisa Shimazaki (M'99) received the B.E. and M.E. degrees in electrical engineering from Nagoya University, Nagoya, Japan, in 1991 and 1993, respectively.

He joined Hitachi, Ltd., Tokyo, Japan, as a VLSI Circuit Engineer. From 2000 to 2001, he was a Visiting Scholar with the University of California at Berkeley, Berkeley, CA, USA, where he researched a high-speed and energy-efficient VLSI circuit. Since 2003, he has been with Renesas Electronics Corporation, where he has been involved in developing low-power digital circuit technique. Currently, he is responsible for the research and development of CPU core and hardware security technology.

Mr. Shimazaki has been a member of the International Technical Program Committee in ISSCC since 2013.

Jun Matsushima received the Foundation degree from the National Institute of Technology, Nara College, Nara, Japan, in 1988.

He joined the General Purpose Computer Division, Hitachi, Ltd., Tokyo, Japan, in 1988. He was with Renesas Technology Corporation, Tokyo, in 2003. Since 2015, he has been with Renesas System Design Co., Ltd., Tokyo, where he is responsible for DFT solution and methodology.

Mr. Matsushima has been a Regular Member of the Information Processing Society of Japan since 1990.