FAULT TREE ANALYSIS OF KNICS RPS SOFTWARE

GEE-YONG PARK[1*], KWANG YONG KOH[2], EUNKYOUNG JEE[2], POONG HYUN SEONG[2], KEE-CHOON KWON[1], and DAE HYUNG LEE[3]
[1]Korea Atomic Energy Research Institute,
1045 Daedeok-daero, Yuseong, Daejeon, 305-353, Republic of Korea
[2]Korea Advanced Institute of Science and Technology,
373-1 Guseong, Yuseong, Daejeon, 305-701, Republic of Korea
[3]Doosan Heavy Industries & Construction,
39-3 Seongbok, Suji, Yongin, Gyeonggi-do, 448-795, Republic of Korea
[*]Corresponding author. E-mail : gypark@kaeri.re.kr

Received July 2, 2007
Accepted for Publication April 12, 2008

This paper describes the application of a software fault tree analysis (FTA) as one of the analysis techniques for a software safety analysis (SSA) at the design phase and its analysis results for the safety-critical software of a digital reactor protection system, which is called the KNICS RPS, being developed in the KNICS (Korea Nuclear Instrumentation & Control Systems) project. The software modules in the design description were represented by function blocks (FBs), and the software FTA was performed based on the well-defined fault tree templates for the FBs. The SSA, which is part of the verification and validation (V&V) activities, was activated at each phase of the software lifecycle for the KNICS RPS. At the design phase, the software HAZOP (Hazard and Operability) and the software FTA were employed in the SSA in such a way that the software HAZOP was performed first and then the software FTA was applied. The software FTA was applied to some critical modules selected from the software HAZOP analysis.

KEYWORDS : Software Safety Analysis, Software Fault Tree Analysis, Digital Reactor Protection System

1. INTRODUCTION

A fully-digitalized reactor protection system, the KNICS RPS, is being developed under the KNICS project for use in newly-constructed nuclear power plants and also in the upgrade of existing analog-based reactor protection systems (RPSs) [1]. The KNICS RPS has four channels which are located in electrically and physically isolated rooms. The KNICS RPS is composed of a group of bistable processors (BPs) which redundantly compare process variables with their corresponding setpoints and a group of coincidence processors (CPs) that generate a trip signal when a trip condition is satisfied by the two-out-of-four voting logic for the trip signals from the BPs. All the trip-actuating functions in the BPs and CPs are implemented in the software.

The software in the KNICS RPS is crucial to the safety of a nuclear power plant in that its malfunction may result in irreversible consequences. The trip-functioning software in the KNICS RPS is thus classified as safety-critical. According to the code and standard [2][3], it is required that an SSA be performed for safety-critical software in the trip-functioning processors. Various standards, including IEC and IEEE standards, have been investigated and compared in order to establish a proper safety analysis process for the safety-critical software in the KNICS project, where not only a digital RPS, but a programmable logic controller (PLC) with a proprietary operating system, which is called a POSAFE-Q PLC, is being developed. Fig.1 shows the safety analysis process in the KNICS RPS and POSAFE-Q PLC. As can be seen in Fig.1, the software HAZOP (Hazard and Operability) is used in the SSA at the requirements phase, and the software HAZOP [4] and the software FTA techniques are used in the design and implementation (code) phases. Among the techniques used in the SSA, this paper describes the application of the software FTA to the safety-critical software of the KNICS RPS at the design phase and presents the analysis results.

2. EFFORTS OF SOFTWARE ERROR REDUCTION AND THE SSA

The software used in the KNICS RPS is being developed under a rigorous procedure [1], and the

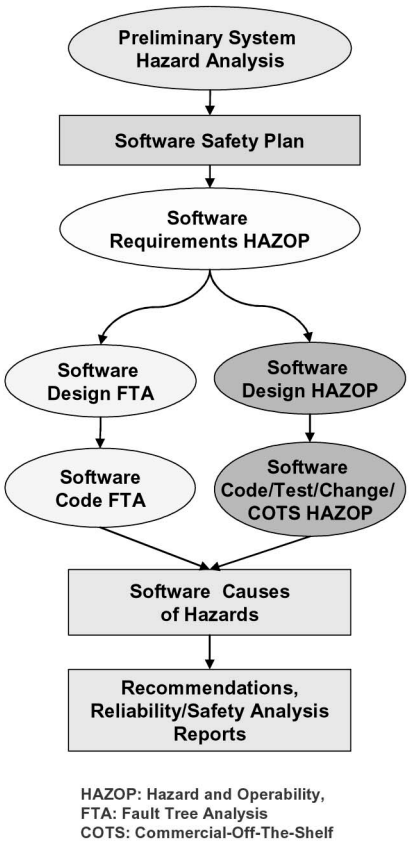


Fig.1 Software Safety Analysis Process for KNICS Project

independent V&V activities are being arranged [5]. Fig.2 shows the V&V activities performed by an independent V&V team for the development of the KNICS RPS software. The purposes of the V&V activities are to ensure that the KNICS software product satisfies the regulatory acceptance criteria and to improve the software quality by finding and resolving software defects at an early phase during software development. For achieving these purposes, various plans are provided and documented at the planning phase. The development of and the V&V activities for the KNICS RPS software are performed according to these plan documents. For the V&V activities during the requirements and design phases, various document evaluations such as the licensing suitability evaluation, the detailed inspection by a Fagan inspection [6], and the traceability analysis are activated. Formal verifications are carried out for the formal specifications by the use of CASE tools. From the implementation phase, the major V&V activity is the software testing. As can be seen in Fig.2, two activities other than the activities described above are presented. One is an SSA, and the other software configuration management. These two activities are combined into the V&V activities in order to satisfy the software quality assurance established in the KNICS project. Thus, the SSA in the KNICS project is part of the V&V activities, as shown in Fig.2.

There are usually four categories of technical methods for achieving highly dependable software: fault

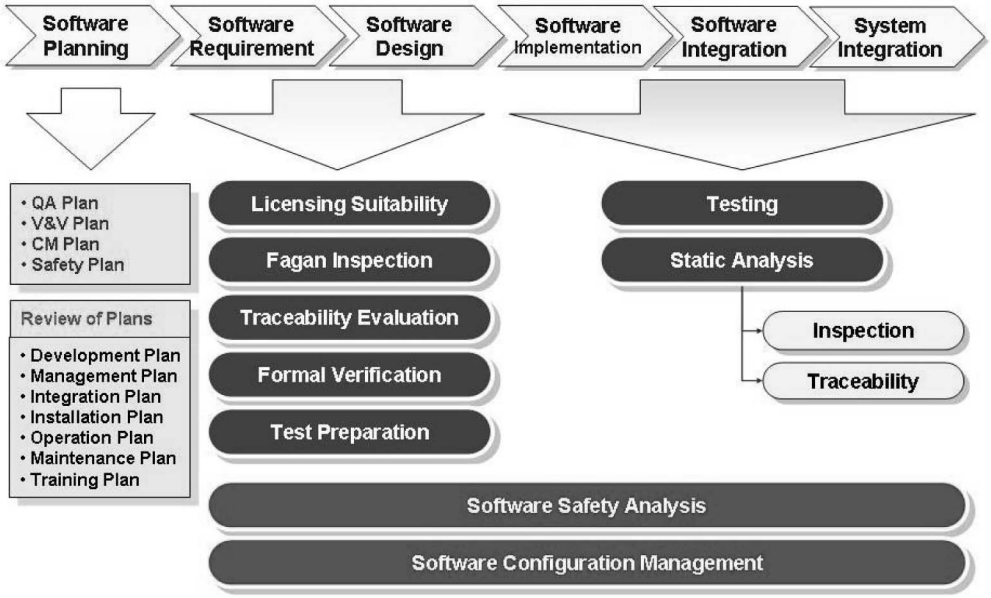


Fig.2 Software V/V Activities of KNICS RPS

prevention, fault removal, fault tolerance, and fault (or failure) prediction [7]. Fault prevention is a means for preventing a fault occurrence or introduction. It contains the use of good software design methods, the enforcement of a structured programming discipline, the employment of formal methods, and so forth. In the development of the KNICS RPS software, the formal specifications/modularizations based on the proprietary CASE tools [8] and some design/coding guidelines are allocated to this category. Fault removal is a means for reducing the presence of faults. The V&V activities and the SSA in Fig.2 are included in this category. Fault tolerance is for ensuring a service capable of fulfilling a system's function in the presence of faults and, to a lesser extent, watchdog timers in the KNICS RPS may be allocated to this category. Fault prediction refers to estimating the present number, future incidence, and consequences of faults. For this category, although numerous methods have been proposed, there are few standard methods with an inter-disciplinary consensus which are applicable to a rare failure event of highly dependable software such as the KNICS RPS software.

3. STRATEGY FOR APPLICATION OF SOFTWARE FTA

As can be seen in Fig.1, the software safety process begins by establishing a software safety plan based on a preliminary system hazard analysis. The SSA activities in the requirements, design, and implementation phases are carried out according to the software safety plan. The purpose of applying the software HAZOP and the software FTA to a software system at the design phase is to identify a defect or hazard in the software modules that can induce or affect the system hazards acquired from a preliminary system hazard analysis or a review of the system-level hazard analysis by an FMEA (Failure Modes and Effects Analysis). For the KNICS RPS, the software-contributable system hazards were identified through a review of the system FMEA results, and they are presented in Table 1. The criticality level in Table 1 is given relative to the severity of a hazard item. Level 4 is the most significant hazard that can drive a plant to a severe accident, and level 1 indicates an insignificant hazard that seldom affects a system's availability.

In an SSA at the design or implementation phase, the software HAZOP was, at first, applied to the software modules represented by a function block diagram (FBD) which is compatible with the POSAFE-Q PLC. The software HAZOP evaluated all the design specifications with respect to all the software-contributable system hazards in Table 1 [4], and the significant defective areas in the FBD modules were identified by this method. The software FTA was then applied to these defective modules to accurately identify a defective location or a logic error. In this study, the software FTA was applied to only the software modules that can induce the first hazard item. Thus, the software FTA is confined to an event where a software module cannot generate a trip signal when a trip condition for the software module is satisfied. Both methods are redundant and complementary in that the software HAZOP is a forward (in fact, HAZOP is a bidirectional method, but, in this study, the forward analysis was weighted more) and broad-thinking analysis method through team works of the HAZOP members, and, on the contrary, the SFTA is a backward and local systematic analysis method by an individual analyst.

Based on the software-contributable system hazards, the interface points between the system hazards and the software modules have been identified. Table 2 presents the software modules of a BP of the KNICS RPS. Actually, all the software modules have been examined to identify the interface points for all the system hazards. In this study, the interface points for hazard item 1 are considered, and the candidate interface points whose final output can affect the most critical system hazard are the trip-functioning modules represented by the bold red lines in Table 2. The software FTA is applied to some of these trip-functioning modules after being determined from the software HAZOP analysis [4], and the interface point through which a defective FBD module can affect the most critical system hazard can be the location of a top event of the software FTA.

Table 1. System Hazards and Criticality Level for KNICS RPS

Item No.	Software Hazards	Criticality Level
1	KNICS RPS cannot generate a trip signal when a trip condition for a process variable is satisfied.	4
2	KNICS RPS generates a trip signal when it should not generate a trip signal.	3
3	KNICS RPS cannot send qualified information of its operational status to the main control room for an operator.	2

Table 2. Software Modules in KNICS RPS BP

NO	Module		Description	OB*	Trip Type*
1	Receive_Signal		HW/SDL/ICN Receive Module		N/A
2	PAT_Scheduler		Automatic Test Scheduler		N/A
3	Test_Selection		Test Selection Module		N/A
4	Trip_ Logic	PZR_PR_Hi Trip	Pressurizer Hi Pressure Trip		Fixed Rising
		SG1_LVL_Lo_RPS Trip	SG-1 Low Level Trip		Fixed Falling
		SG1_LVL_Lo_ESF Trip	SG-1 Low Level Trip for ESF		Fixed Falling
		SG1_LVL_Hi Trip	SG-1 Hi Level Trip		Fixed Rising
		SG1_PR_Lo Trip	SG-1 Low Pressure Trip		MR, Falling
		CMT_PR_Hi Trip	Containment Hi Pressure Trip		Fixed, Rising
		CMT_PR_HH Trip	Containment Hi-Hi Press. Trip		Fixed, Rising
		SG1_FLW_Lo Trip	SG-1 Low Coolant Flow Trip		RR, Falling
		PZR_PR_Lo Trip	Pressurizer Low P Trip	Y	MR, Falling
		VA_OVR_PWR_Hi Trip	Variable Over Power Hi Trip		RR, Rising
		SG2_LVL_Lo_RPS Trip	SG-2 Low Level Trip		Fixed Falling
		SG2_LVL_Lo_ESF Trip	SG-2 Low Level Trip for ESF		Fixed Falling
		SG2_LVL_Hi Trip	SG-2 Hi Level Trip		Fixed Rising
		SG2_PR_Lo Trip	SG-2 Low Pressure Trip		MR, Falling
		SG2_FLW_Lo Trip	SG-2 Low Coolant Flow Trip		RR, Falling
		LOG_PWR_Hi Trip	Log Reactor Power Hi Trip	Y	Fixed Rising
		DNBR_Lo Trip	Low DNBR Trip		Digital
		LPD_Hi Trip	Hi LPD Trip		Digital
		CPC_CWP Trip	CPC_CWP		Digital
5	Test_Results_Handler		Test Results Handling Module		N/A
6	HB_MONITORING		Heartbeat Monitoring Module		N/A
7	HB_Gen		Heartbeat Generation Module		N/A
8	Ch_Bypass_Send_Receive		Channel Bypass Transfer Module		N/A
9	Send_Signal		HW/SDL/ICN Sending Module		N/A

* Fixed: Fixed Trip Setpoint; MR: Variable Trip Setpoint by Manual Reset; RR: Variable Trip Setpoint by Automatic Rate-Limiting; Digital: ON/OFF Trip Signal; OB: Operating Bypass

4. APPLICATION OF SOFTWARE FTA

As supposed by Leveson, Cha, and Shimeall [9], the purposes of the software FTA are to detect software logic errors, to determine the conditions under which fault-tolerance and fail-safe procedures should be initiated, and to facilitate effective safety testing by pinpointing critical functions and test cases. The FTA is a well-established safety analysis technique in nuclear power plants [10], and it has been widely used in the safety analysis. Safety analysis by the FTA for software is slightly different from that for process systems, where a fault event of a system component is based on a probabilistic nature, because of the fact that the software is configured based on the logistic constructs and its behavior is deterministic. Therefore, the software FTA has usually been constructed based on the fault tree templates for a code. At the design phase of the KNICS RPS software, the detailed design descriptions were presented by the FBD modules. The software FTA based on fault tree templates for the FBs is

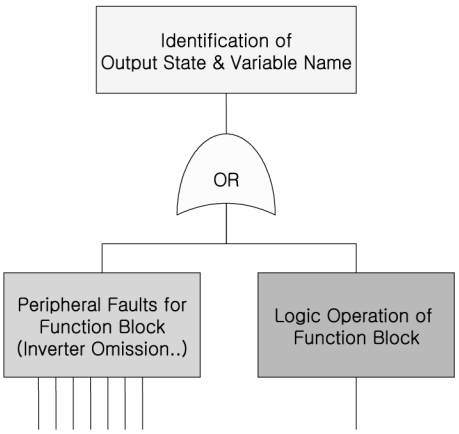


Fig.3 Overall Architecture for Constructing Fault Tree Templates for Function Blocks

applied to a part of the software modules determined from the software HAZOP, and the top node is only related to the most safety-critical hazard.

4.1 Construction of Software Fault Tree Templates

The fault tree templates are actually small fault trees for their corresponding components in the software, and one more different aspect of the software FTA is that an event in a fault tree template may be a logic operation, which is prohibited in the conventional FTA since all the events that are linked together on a fault tree should be written as faults [10]. For a typical FB in the FBD module, a fault tree template is constructed in a way that the failure modes are extracted starting from the output port of an FB, through the body of the FB, ending at the input ports, as shown in Fig.3. The lower left event in Fig.3 indicates plausible physical and functional faults within an FB, and the lower right event is for a logic operation through which a template at the immediate lower tree level is attached to its upper-level template.

The fault tree templates for the FBs have been proposed by Oh and her colleagues [11]. Oh [12] classified fault cases rigorously and derived each FB's template based on these fault cases. The fault tree templates are composed of two distinguishing parts: one is the fault events and the other the cause/effect events. One purpose for the introduction of the cause/effect events is to indicate fault propagation and also to help an analyst understand the logical operation of a function block. From the experience of applying the fault tree templates to the safety-critical software of the KNICS RPS, it was found that, before the construction of a software FTA, analysts usually reviewed in advance the function block diagram in great detail and they were inclined to focus more on fault/failure cases because they already understood the logical flow of an FBD module. Thus, in this study, the templates for the FBs were refined to be more fault-oriented and concise.

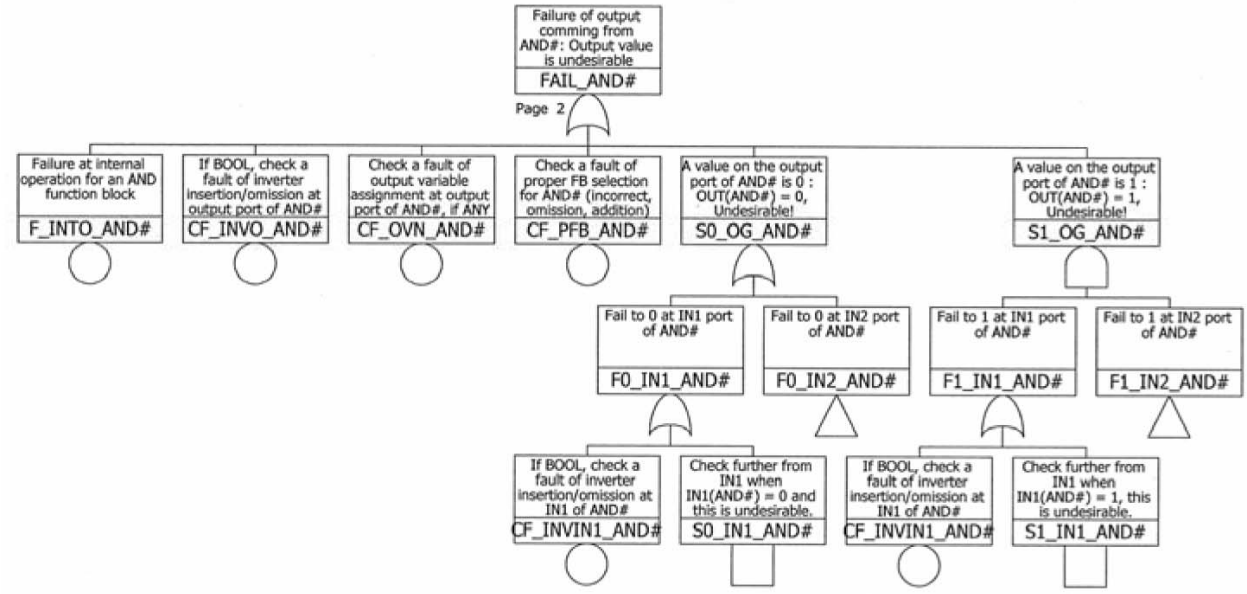


Fig.4 Fault Tree Template for the AND Function Block

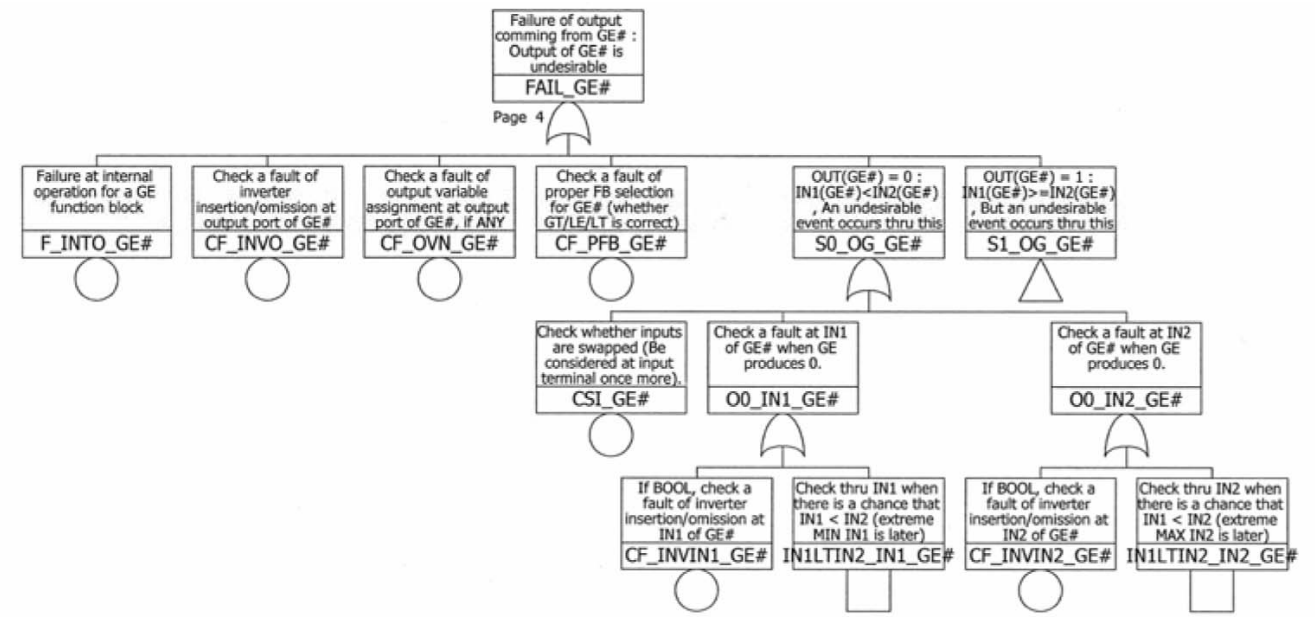


Fig.5 Fault Tree Template for the GE Function Block

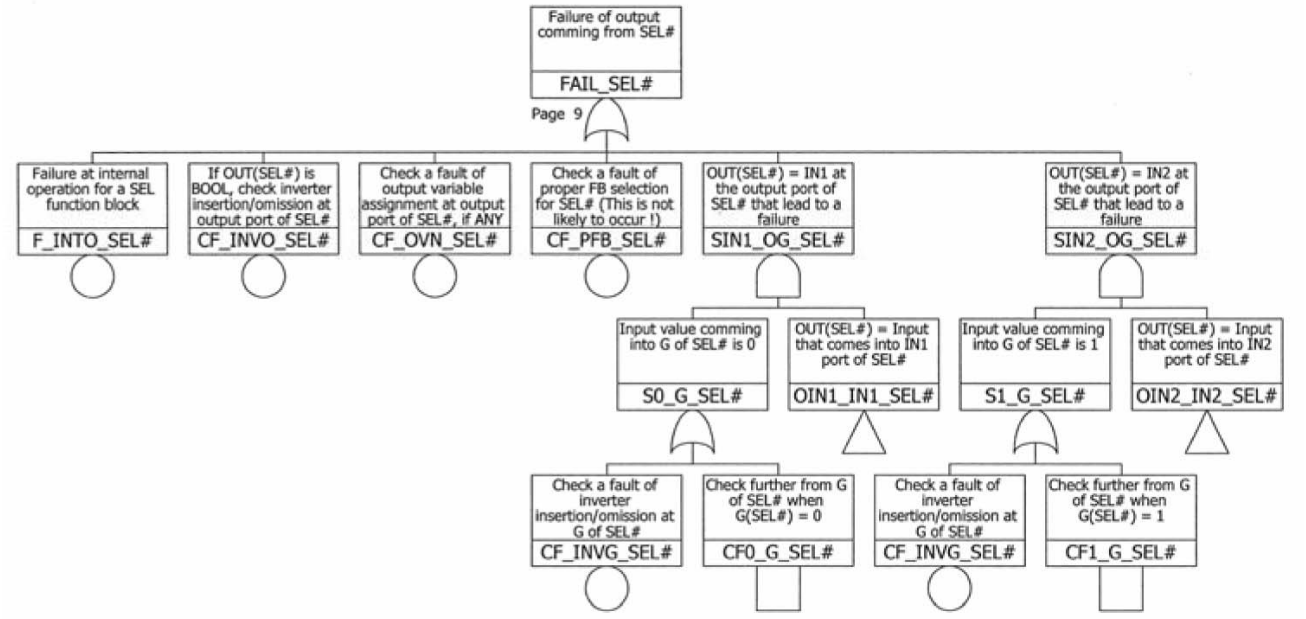


Fig.6 Fault Tree Template for the SEL Function Block

The types of the FBs used in the FBD modules are divided into five classes: Logic Operation FB (AND/OR), Comparison FB (GE/GT/LE/LT/EQ), Selection FB (SEL), Algebraic Operation FB (ADD/SUB/MUL/DIV/ABS), and Timer FB (TON). The TON is activated in such a way that if the IN value is 1 and the PT value is set to an appropriate value, then the output of the TON becomes 1 when the internal count is equal to or larger than the PT value. Figs.4-8 display the modified fault tree templates for the representative FBs for the five function types. The fault events in a fault tree template are derived based on the fault criteria proposed by Oh [12]. In Figs.4-8, the

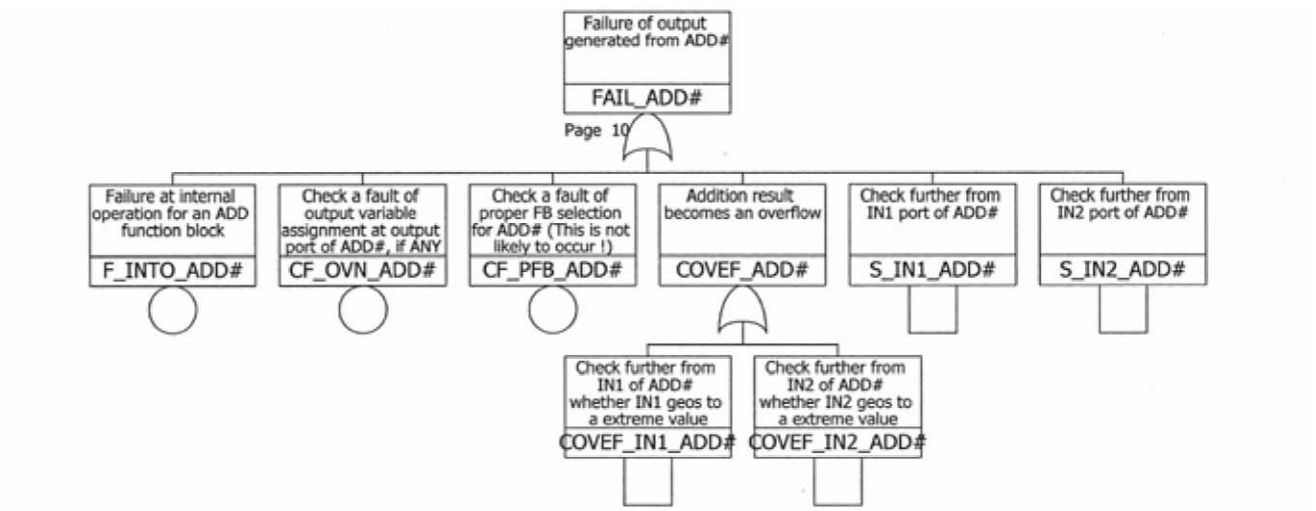


Fig.7 Fault Tree Template for the ADD Function Block

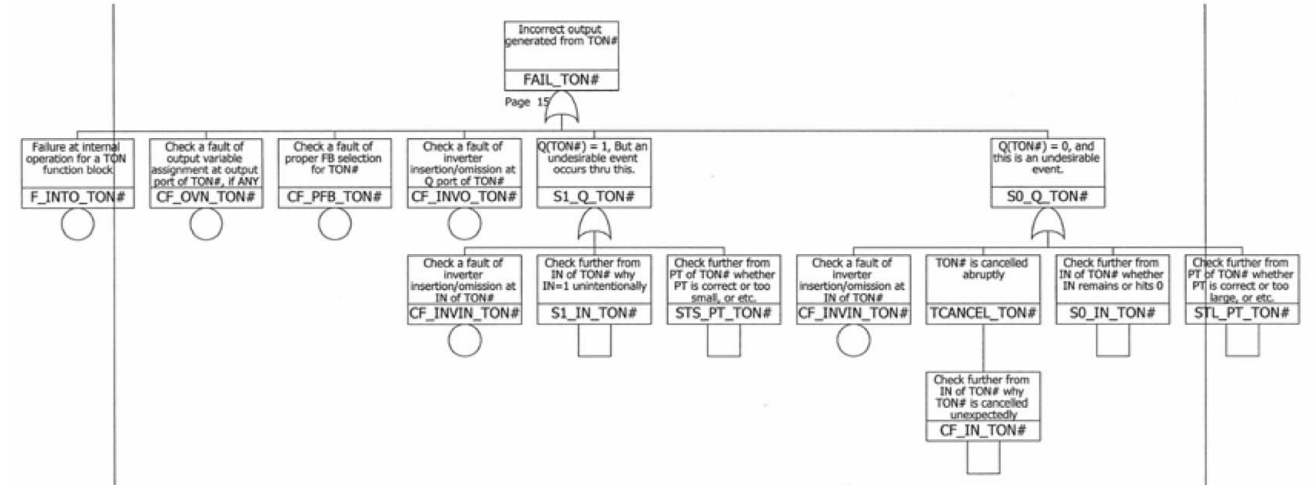


Fig.8 Fault Tree Template for the TON Function Block

box with a circle at the bottom means a basic event, the box with a triangle represents an event that has more trees presented on another page, and the one with a rectangle indicates that a further analysis for a lower-level tree could progress through this event. An OR- or AND-gate symbol below a box performs a logical OR or AND operation for its inputs.

4.2 Software FTA

The software modules selected from the results of the software HAZOP for the BP FBD modules in Table 2 are SG1_FLW_Lo Trip (Steam Generator #1 Low Coolant Flow Trip), PZR_PR_LO Trip (Pressurizer Low Pressure Trip), VA_OVR_PWR_Hi Trip (Variable Over-Power High Trip), and DNBR_Lo Trip (Low DNBR Trip). To demonstrate the results of the software FTA, the safety analysis for the trip module of VA_OVR_PWR_Hi Trip is presented in this paper. The VA_OVR_PWR_Hi Trip module generates a trip signal for the shutdown of a nuclear reactor when the neutron flux is increasing with a rate of change larger than the acceptable rate at a low-power startup time or the value of the neutron flux exceeds the maximum limit at the rated power. It is further composed of sub-modules such as TRIP_DECISION, TRIP_OPERATION, SETPT_CAL, and TEST_SEL. Among the sub-modules in the VA_OVR_ PWR_Hi Trip module, the TRIP_OPERATION, which is a major sub-module, is selected for the application of the software FTA.

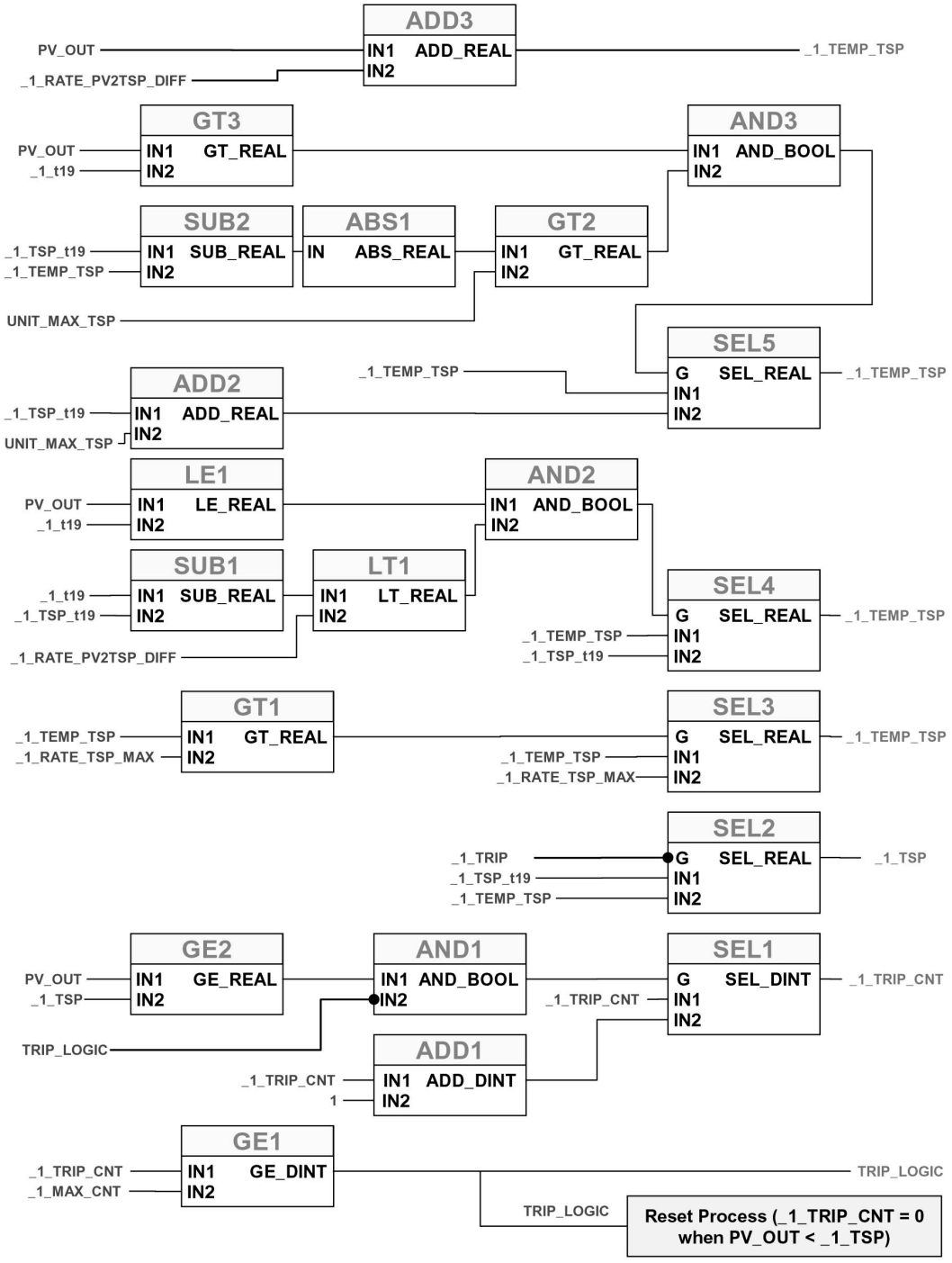
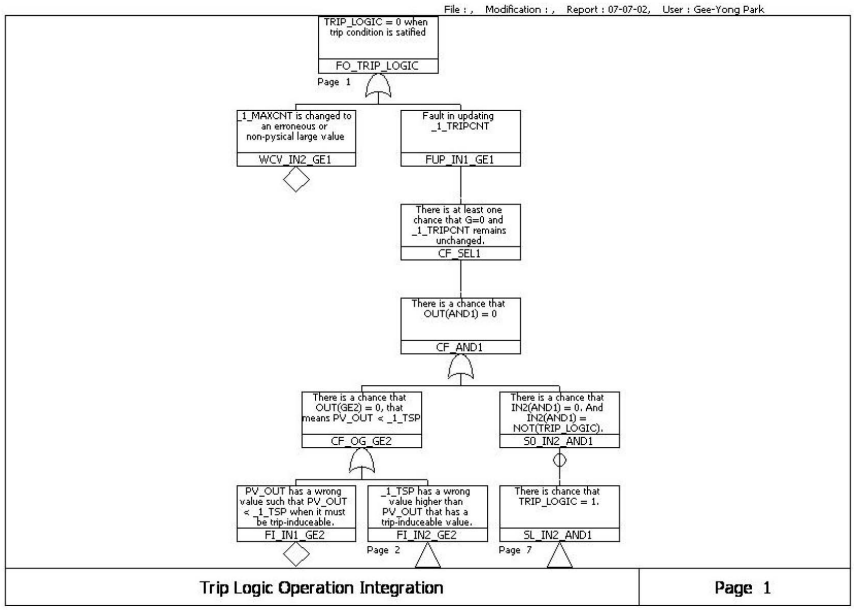


Fig.9. FBD Module for VA_OVR_PWR_Hi Trip (TRIP_OPERATION)

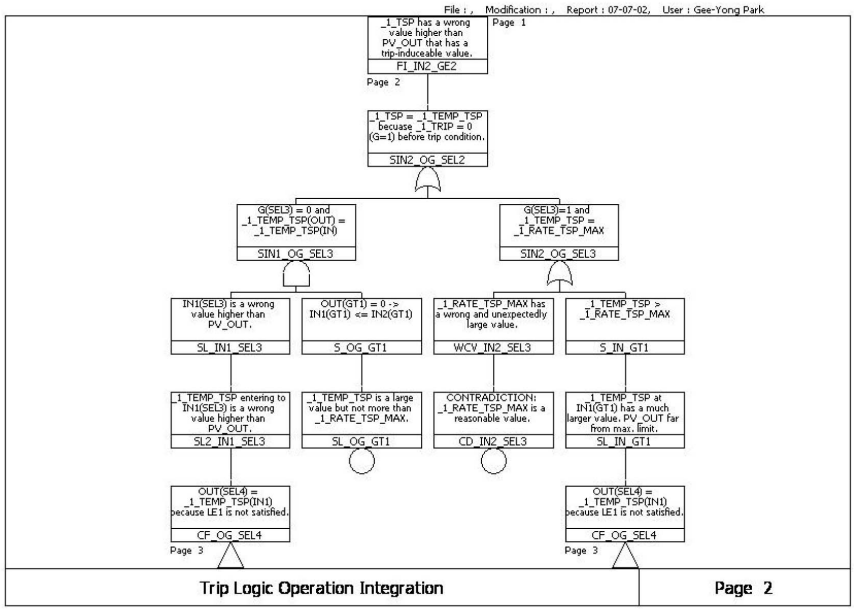
A part of the FBD representation for the TRIP_OPERATION sub-module is shown in Fig.9 where the function blocks are labeled, such as GE1. The flow path (i.e., the sequence of execution) in the FBD modules in Fig.9 is from left to right and from top to bottom. For these FBD modules, the software FTA based on the fault tree templates are constructed as in Figs.10(a)-(i), where the software FTA are pruned to leave meaningful trees. In Figs.10, an event box with a diamond symbol attached below it indicates that the

event is not analyzed further because of a lack of information or inappropriateness in doing so. An event box with a house symbol means that the event is natural.

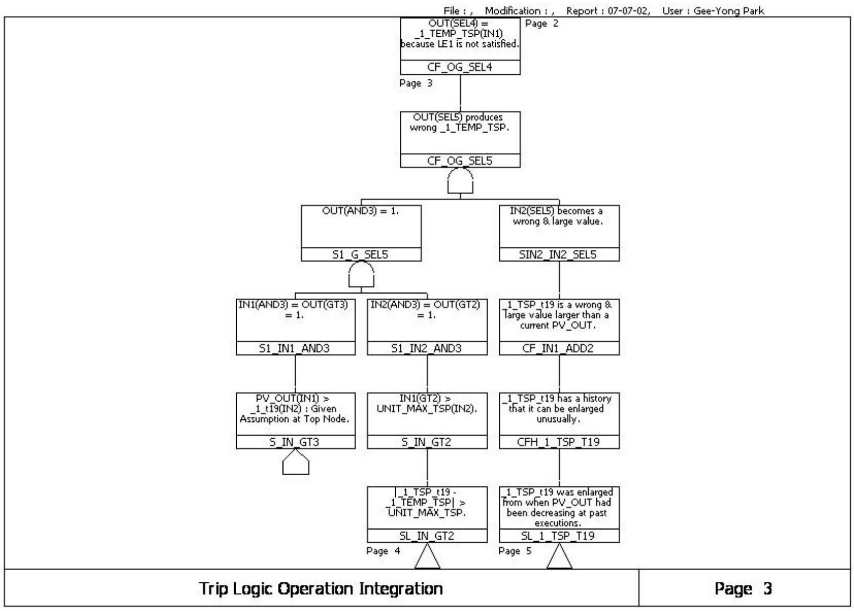
For the TRIP_OPERATION sub-module, the final output is the variable TRIP_LOGIC at the output port of GE1 in Fig.9. When a trip condition is satisfied, this variable should be 1. Thus, the top event of the software FTA in Fig.10(a) is the event that TRIP_LOGIC=0 when a trip condition is satisfied in such a way that PV_OUT is larger than _1_TSP (at GE2) with an internal count _1_TRIP_CNT being equal to or larger than a constant _1_MAXCNT (at GE1). It is apparent that this event is due to an incorrect operation in GE1, leading to the conclusion that the input variable _1_TRIP_CNT of GE1 has a problem with its updating procedure, as shown in the event FUP_IN1_GE1 in Fig.10(a). In the lower part of Fig.10(a), this problem is traced to AND1 and GE2 in Fig.9, and this fault back-tracking is divided into two



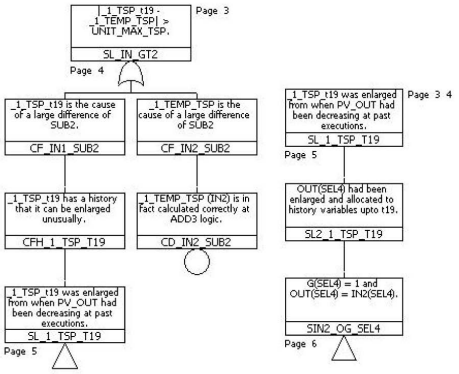
(a) Page 1



(b) Page 2

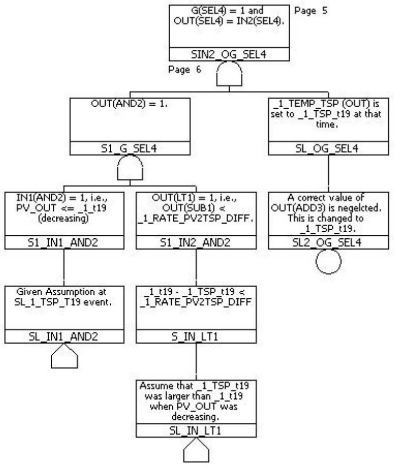


(c) Page 3

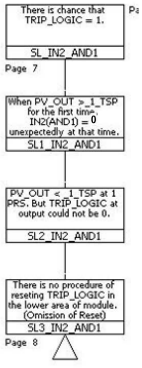


(d) Page 4

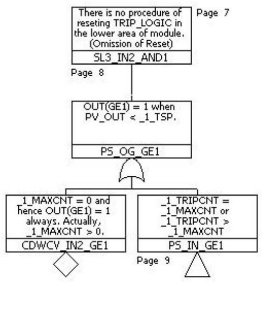
(e) Page 5



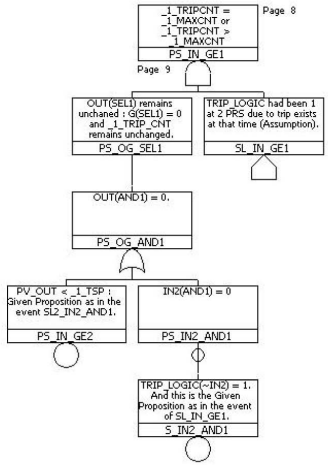
(f) Page 6



(g) Page 7



(h) Page 8



(i) Page 9

(Note: PRS - Previous Run Step)

Fig.10 Software FTA for VA_OVR_PWR_Hi Trip

paths: one is due to an incorrect calculation of the trip setpoint _1_TSP at GE2 and the other due to an incorrect value of the input variable TRIP_LOGIC at AND1. The first path is further arranged in Figs.10(b)-(f).

From Figs.10(b)-(f), it can be recognized that, when a process variable PV_OUT is decreasing, the interim trip setpoint _1_TEMP_TSP at SEL4 is set to _1_TSP_t19 rather than to _1_TEMP_TSP at IN1 of SEL4 though _1_TEMP_TSP at IN1 contains the correct trip setpoint. This fault results from the wrong operation between SUB1 and LT1 in Fig.9. Fig.10(f) reveals a defect in the operational logic (i.e., for normal conditions, the operation that _1_t19 - _1_TSP_t19 usually results in a negative value) between SUB1 and LT1. The above identification means that, when the process variable decreases, the trip setpoint remains unchanged with a value of _1_TSP_t19 (the trip setpoint at 1 second before). Thus, the difference between the trip setpoint and the process variable becomes increasingly larger when the process variable continuously decreases. After a decreasing trend, if the process variable starts to increase so fast that it can trigger the expected and correct trip setpoint, the trip signal cannot be generated because the trip setpoint before an increasing trend has been given to a much larger value than an expected and correct value.

The second fault path in Figs.10(g)-(i) is plausible in the situation that a process variable has increased to generate a trip signal (i.e., _1_TRIP = 1) and then it decreased slightly to a value lower than the trip setpoint, but, at this time, the process variable increases suddenly to a value above the trip setpoint. Figs.10(g)-(i) reveal that this phenomena can occur due to an incomplete reset procedure of TRIP_LOGIC resulting from a defect in the reset logic of _1_TRIP_CNT. The reset process of _1_TRIP_CNT, as shown in the bottom right area of Fig.9, means that _1_TRIP_CNT becomes 0 when both PV_OUT < _1_TSP and TRIP_LOGIC = 1 are satisfied, causing TRIP_LOGIC to be reset to 0 one step later when PV_OUT < _1_TSP. Thus, when the software FTA is confined to the TRIP_OPERATION sub-module, the trip triggering finally occurs with a one step delay. Generating a trig signal with a one step delay may violate the system response time for the KNICS RPS.

As can be seen in Figs.10, the software FTA for a part of the VA_OVR_PWR_Hi Trip module has a complex and lengthy tree structure where the software HAZOP seemed to be impossible to apply to pinpoint a defect. The logic errors described above have not been detected even in a formal verification process. Though a testing may identify these errors, it is very difficult to elucidate these types of defects without delicate test cases with a profound test scenario. The results of the software FTA could be used in providing delicate test cases for identifying defects containing these types of logic errors.

5. RESULTS

For the safety analysis of the safety-critical software, the strategy and application procedure for the software FTA were presented in this paper. Based on the previous study of the fault tree templates for the function blocks, the fault-oriented templates were devised for the convenient implementation of the software FTA. Because the software fault trees for an FBD module are usually very long and complex, the software FTA is applied to critical portions of the FBD-based design modules that are identified from the software HAZOP. Because of a different viewpoint from the V&V activities, the software FTA can obtain some valuable results that have not been identified through a rigorous V&V procedure.

In the KNICS RPS software, the software HAZOP and the software FTA are used in the SSA at the design phase. The application of both methods is supposed to be redundant, and this redundancy obviously requires an additional, but overlapping, work for the SSA. This type of overlap is thought to be meaningful because all the current safety analysis methods have their own merits and demerits.

REFERENCES

[1] J. H. Park, D. Y. Lee, C. H. Kim, "Development of KNICS RPS Prototype", *Proceedings of ISOFIC 2005*, **Session 6**, pp.160-161, Tongyeong, Korea, Nov. 1~4, 2005.

[2] NUREG-0800, Rev.04, "Standard Review Plan: BTP HICB–14, Guidance on Software Reviews for Digital Computer-Based Instrumentation and Control Systems," U.S. Nuclear Regulatory Commission, 1997.

[3] IEEE Std-1228, "Software Safety Plan", Institute of Electrical and Electronic Engineers, 1994.

[4] G. Y. Park, J. S. Lee, S. W. Cheon, K. C. Kwon, E. Jee, and K. Y. Koh, "Safety Analysis of Safety-Critical Software for Nuclear Digital Protection System", Lecture Notes in Computer Science, **Vol.4680**, pp.148-161, 2007.

[5] K. C. Kwon and G. Y. Park, "Formal Verification and Validation of the Safety-Critical Software in Digital Reactor Protection System", *NPIC & HMIT 2006*, pp.1371-1376, Nov. 12~16, Albuquerque, NM, USA, 2006.

[6] M. E. Fagan, "Design and Code Inspections to Reduce Errors in Program Development", *IBM System Journal*, **Vol.15**, No.3, pp.182-211, 1976.

[7] M. R. Lyu, *Handbook of Software Reliability Engineering*, McGraw-Hill, pp.29, 1996.

[8] J. Yoo and S. Cha, "A Formal Software Requirements Specification Method for Digital Plants Protection Systems", CS/TR 2003-191, Department of Computer Science, KAIST, 2003.

[9] N. G. Leveson, S. Cha, and T. J. Shimeall, "Safety Verification of Ada Programs using Software Fault Trees," *IEEE Software*, pp.48-59, July 1991.

[10] W. E. Vesely, F. F. Goldberg, N. H. Reberts, and D. F. Haasl, *Fault Tree Handbook*, NUREG-C492, U. S. Nuclear Regulatory Commission, 1981.

[11] Y. Oh, J. Yoo, S. Cha, and H. S. Son, "Software Safety Analysis of Function Block Diagrams using Fault Trees", *Reliability Engineering and System Safety*, **Vol.88**, pp.215-228, 2005.

[12] Y. Oh, *Safety Analysis of Function Block Diagrams using Fault Trees*, M.S. Thesis, EECS Department, Korea Advanced Institute of Science and Technology, Korea, 2004.