Effect analysis of faults in digital I&C systems of nuclear power plants

LEE Seung Jun[1]

1. Integrated Safety Assessment Division, Korea Atomic Energy Research Institute, 1045 Daedeok-daero, Yuseong-gu, Daejeon, Korea (sjlee@kaeri.re.kr)

Abstract: A reliability analysis of digital instrumentation and control (I&C) systems in nuclear power plants has been introduced as one of the important elements of a probabilistic safety assessment because of the unique characteristics of digital I&C systems. Digital I&C systems have various features distinguishable from those of analog I&C systems such as software and fault-tolerant techniques. In this work, the faults in a digital I&C system were analyzed and a model for representing the effects of the faults was developed. First, the effects of the faults in a system were analyzed using fault injection experiments. A software-implemented fault injection technique in which faults can be injected into the memory was used based on the assumption that all faults in a system are reflected in the faults in the memory. In the experiments, the effect of a fault on the system output was observed. In addition, the success or failure in detecting the fault by fault-tolerant functions included in the system was identified. Second, a fault tree model for representing that a fault is propagated to the system output was developed. With the model, it can be identified how a fault is propagated to the output or why a fault is not detected by fault-tolerant techniques. Based on the analysis results of the proposed method, it is possible to not only evaluate the system reliability but also identify weak points of fault-tolerant techniques by identifying undetected faults. The results can be reflected in the designs to improve the capability of fault-tolerant techniques.

Keyword: digital I&C; fault-tolerant technique; fault injection

1 Introduction

Instrumentation and control (I&C) are important in reliability analysis of nuclear power plants (NPPs). Operators in a plant are provided the plant information and perform required controls through I&C systems. Also, safety systems such as a reactor protection system (RPS) automatically generate a reactor trip signal when trip parameters are over the trip set points.

Recently, I&C systems have undergone digitalization. Deterioration and an inadequate supply of components of analog I&C systems have led to inefficient and costly maintenance. Moreover, since the fast evolution of digital technology has enabled more reliable functions to be designed for NPP safety, the transition from analog to digital has been accelerated.

The unique features of digital I&C systems can be advantageous for plant safety by providing useful functions such as fault-tolerant techniques, but they can have negative effects caused by uncertain reliability such as software. Therefore, a reliability analysis of digital systems has been introduced as one of the important elements of a probabilistic safety assessment (PSA)[1-4].

One of the most significant features of digital I&C systems is the fault-tolerant technique. The fault detection coverage of the fault-tolerant techniques must be considered in the safety evaluation of a system because faults in the system do not have effect on the safety if they are detected by fault-tolerant techniques. If a system has ideally perfect fault detection coverage, the faults in the system can be ignored in the safety evaluation. Therefore, it is important to evaluate the fault detection coverage of the fault-tolerant techniques, to analyze the propagation path of these faults to failures, and to identify the root causes of undetected faults.

In this work, the fault detection coverage of a digital I&C system was evaluated by fault injection experiments. And the undetected faults were analyzed and a model to represent the propagation path of the faults to the failures was developed.

Received date: September 12, 2014

Using the model, it can be identified how a fault is propagated to the output and why a fault is not detected by fault-tolerant techniques.

2 Fault-tolerant techniques

Digital I&C systems are designed using various types of fault detection functions for fault-tolerance. Although fault detection functions aim to enhance the safety by detecting faults in a system and eliminate the negative effect of such faults, more fault detection functions do not necessarily ensure higher fault detection coverage. For example, if a system is inspected by three fault-tolerant techniques as shown in Fig. 1, some faults are detected by only one fault detection function, and some are detected by two or more fault detection functions. Some faults are not detected by any fault detection functions. The overall fault coverage of fault-tolerant techniques implemented in the system is not the simple summation of fault coverage of each fault-tolerant technique, but a union set of fault coverage of all the fault-tolerant techniques[1].

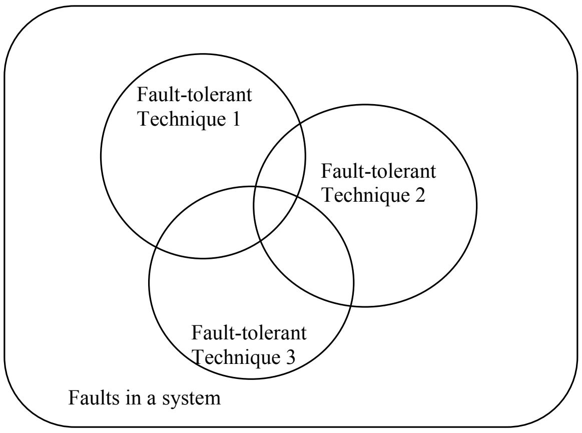

Fault-tolerant Technique 1
Fault-tolerant Technique 2
Fault-tolerant Technique 3
Faults in a system

Fig. 1 Fault-tolerant techniques and their coverage.

The total average unavailability q_T for periodically tested components is calculated based on equation (1) [1].

$$q_T = \lambda T / 2 + \lambda T_R \tag{1}$$

where,

λ = Failure rate for the portion detected by a fault-tolerant technique (or fault-tolerant techniques) in the component

T = Time interval of the fault-tolerant technique in the component

T_R = Time required for maintenance

If we assume that a system checks its availability through periodic fault-tolerant techniques and manual tests and that manual test detects all the faults which are not detected by other fault-tolerant techniques, then the unavailability of the system could be calculated using the following equation.

$$Q = \sum_{i=1}^{n} \lambda_i \left(\frac{T_i}{2} + T_R\right) + \lambda_M \left(\frac{T_M}{2} + T_R\right) \tag{2}$$

where,

Q = System unavailability

n = number of areas

λ_i = Failure rate for the portion detected by a fault-tolerant technique (or fault-tolerant techniques) in area i

λ_M = Failure rate for the portion not detected by any fault-tolerant techniques

T_i = Time interval of the fault-tolerant technique in area i

T_M = Time interval of the manual test

T_R = Time required for maintenance

And the failure rates of each area are represented using the failure rate of the system and the fault detection coverage of each area.

$$\lambda_i = \lambda \cdot C_i \tag{3}$$

$$\lambda_M = \lambda\left(1 - \sum_{i=1}^{n} C_i\right) \tag{4}$$

where,

λ = Failure rate of the system

C_i = fault detection coverage of a fault-tolerant technique (or fault-tolerant techniques) in area i

Since components are inspected by different fault-tolerant techniques with different inspection periods, the specific fault detection coverage for a component should be identified for accurate unavailability evaluation.

3 Fault tnjection experiments

In this work, fault injection experiments were performed to identify the fault detection coverage of a system and to analyze the effects of the faults in the system. Based on the experiment results, the undetected faults were analyzed to identify root causes.

3.1 Target system

For a more realistic evaluation, the prototype of a digital reactor protection system that have been adopted in a real digitalized NPP were used for the experiment. The target digital I&C system is the Integrated Digital Protection System (IDiPS) Reactor Protection System (RPS), which was developed in Korea[5,6] during the Korea Nuclear Instrumentation and Control System (KNICS) research and development project. The IDiPS RPS has four independent channels, where each channel consists of bistable processors (BPs), coincidence processors (CPs), an automatic test and interface processor (ATIP), a cabinet operator module (COM), and other hardware components[7].

IDiPS RPS tests are classified into two categories: active tests and passive tests. Figure 2 shows four types of tests that have different types of coverage and periods[7].

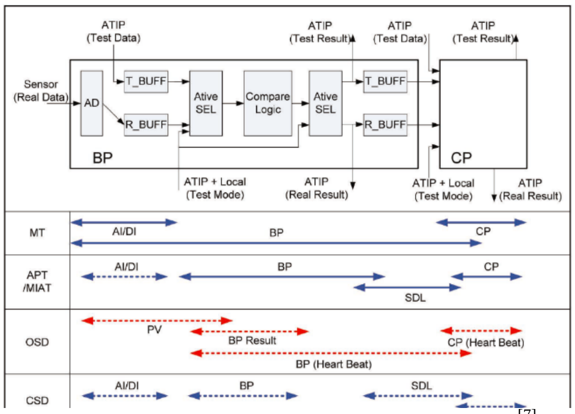


Fig. 2 Fault-tolerant techniques in the IDiPS RPS [7]

- Active tests consist of automatic periodic tests (APTs), manual initiated automatic tests (MIATs), and manual tests (MTs) (Hur *et al.*, 2009). An APT is periodically initiated by the ATIP without any human intervention. An MIAT is almost the same as an APT except for the operator initiation and tested trip parameter selection. An MT is generally performed once per month.
- A passive test partially checks the system's integrity. This test consists of component self-diagnostics (CSD) and online status diagnostics (OSD).

3.2 Fault injection methods

Fault injection is a technique for validating the reliability of a fault-tolerant system. It consists of controlled experiments where the observation of the system's behavior in the presence of faults is explicitly induced. Fault injection techniques can be classified into three main categories[8,9]:

(1) Hardware-implemented fault injection: This is accomplished at the physical level by disturbing the hardware with parameters of the environment (heavy ion radiation, electromagnetic interferences, *etc.*) or by modifying the value of the integrated circuit pins.

(2) Software-implemented fault injection: The objective of this technique is to reproduce at the software level errors that would have been produced upon the occurrence of faults in either hardware or software. This is based on different practical types of injection, including the modification of memory data, the mutation of application software, and the lowest service layers (for example, at the operating system level).

(3) Simulated fault injection: In this technique, the system undergoing testing is simulated in another computer system. Faults are induced, altering the logical values of the model elements during the simulation.

To identify the exact fault detection coverage, the best method is to simulate all possible faults physically by using hardware-implemented fault injections. However, it is difficult to simulate all faults by using hardware-implemented fault injection techniques because this requires expensive hardware, and some faults cannot be controlled and are limited owing to the complexity of the system [1]. Therefore, we used a software-implemented fault injection technique in which faults can be injected only into the memory. Our fault injection experiment was conducted based on the assumption that all faults in a system are reflected in the faults in the memory because a fault should affect the memory related to the calculation process or variables for causing a wrong output. A fault of any component in a system may have an effect on the calculation process, reading input variables, generating output variables, and so on. A wrong calculation, program halt, variable changes, or wrong execution path may be caused by the fault. If a fault does not have any effect on the output, then it is impossible to detect the fault because there are no observable consequences from the fault.

3.3 Fault types

Faults in digital I&C systems are categorized into seven types according to their consequence and detection potential, as shown in Table 1[10].

Table 1 Categorization of faults into seven types

	Changed and used			Unused or unchanged
	Correct output	Wrong output	No output	
Detected	A	C	E	G
Undetected	B	D	F	

Correct output (Fault types A and B):

- Even when a bit is changed by a fault and the changed bit is used to generate a system output, there may not be any effect on the output because the changed bit is not directly related to the output generation. For example, a stuck-at-1 fault changes “variable A” from 16 (binary: 10000) to 24 (binary: 11000). In this case, if the set point for “variable A” is 10, then the output is not changed, because both 16 and 24 are greater than the set point. This type of fault is categorized as a safe fault.

Wrong output (fault types C and D):

- The bit changed by a fault may cause a wrong system output. For example, “variable A” has a value of 16 (binary: 10000), and the set point is 10. If the highest bit of “variable A” is changed by a stuck-at-0 fault, then “variable A” becomes 0 (binary: 00000) and a wrong output is generated.

No output (fault types E and F):

- The bit changed by a fault may cause a program halt or infinite loop, and thus the program does not generate an output. In this case, nothing is written on bits for the output, and the previous output is not updated.

Unused or unchanged (fault type G):

- A memory area is not assigned to any program code or variables. Even though some memory area is assigned and used, there will be no effect on the output unless a fault changes a memory bit. For instance, if a stuck-at-0 fault is injected on a bit that was already 0, then nothing is changed. These unused or unchanged bits do not have any effect on the output generation, and it is impossible to detect such faults.

If a system works correctly despite the presence of a fault, the fault is called a “safe fault.” A “correct output” (fault types A and B) and “unused or unchanged” (fault type G) fault types are classified as a “safe fault.” Even if a malicious fault causes a “wrong output” or “no output” (fault types C, D, E, and F), if it is detected, the system will remain in a safe state. Such detectable malicious faults (fault types C and E) are also classified as a “safe fault” in terms of safety. If a malicious fault is not detected, the fault is classified as an “unsafe fault.” The fault types are categorized as shown below:

- No-effect faults: A, B, G
- Malicious faults: C, D, E, F
- Safe faults: A, B, C, E, G
- Unsafe faults: D, F

In terms of safety, and from the viewpoint of PSA, important factors are malicious faults and their detection probability. Even though there are faults and something is changed by the faults, it does not matter if a trip signal is generated. A dangerous situation is when a trip signal is not generated and no fault-tolerant techniques can detect the abnormal behavior of the digital I&C systems when a reactor should be tripped. Therefore, the fault detection coverage is defined as the probability of detecting malicious faults as follows:

$$\text{Fault detection coverage} = \frac{C+E}{C+D+E+F} \qquad (5)$$

3.4 Fault injection experiment results

We performed fault injection experiments on the memory area of the BP application. Faults were injected into the memory of the BP application by using the Code Composer tool[11], and an automatic fault injection program was developed for the experiment. Fig. 3 shows the environment of the fault injection experiment. Two types of memory faults, stuck-at-0 and stuck-at-1, were considered because a memory bit has a binary value. In the experiment system, three fault-tolerant techniques are available: OSC, CSD, and APT.

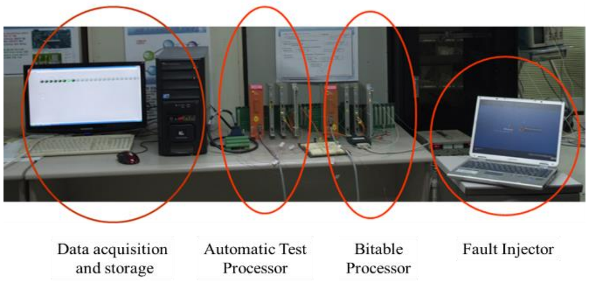


Fig. 3 The environment for the fault injection experiment [7].

A total of 55,752 fault injection experiments were performed and the following observations were made. Fig. 4 shows the number of faults detected by three fault-tolerant techniques.

- Faults resulting in no effect (fault types A, B, and G): 90.77% of injected faults
- Faults resulting in no trip (fault types C, D, E, and F): 5,144 (9.23% of injected faults)
- Detected faults (fault types C and E): 5,028 (9.02% of injected faults)
- Undetected faults (fault types D and F): 116 (0.21% of injected faults)

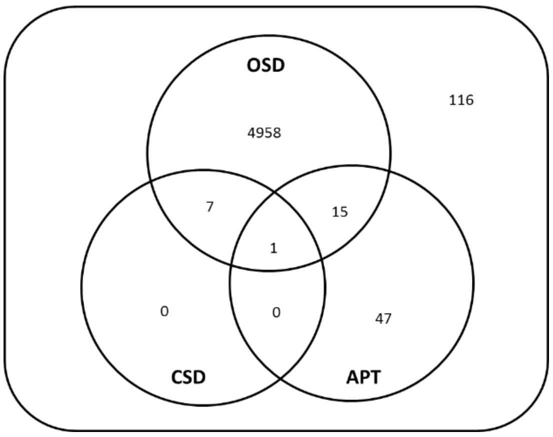


Fig. 4 The number of faults covered by fault detection functions.

Among the faults that caused a trip signal generation failure (C + E + D + F), the undetected faults (D + F) occupied 2.26%. Therefore, the fault detection coverage of the target system was 97.74%, based on equation (5).

4 Analysis of undetected faults

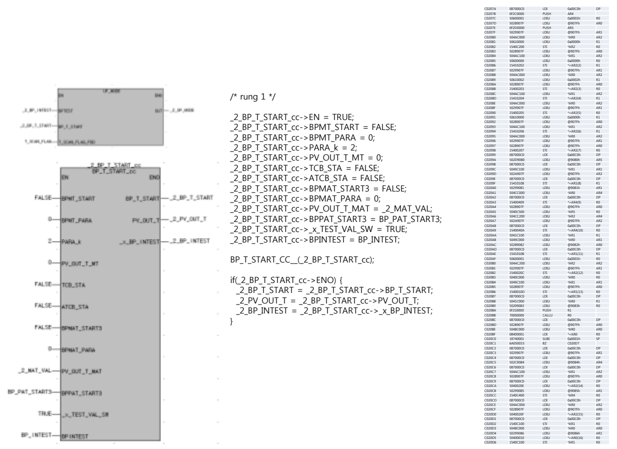
To improve the fault detection coverage, it is necessary to analyze undetected faults by fault-tolerant techniques. As shown in the experiment result, 116 faults are undetected and cause no-trip when trip is required (fault types D and F). A detailed analysis of them was performed.

In fault injection experiments, only the consequence of a fault can be observed. This is a kind of block box test in which internal processing is not considered. However, to identify the reasons of the undetected malicious faults, it is necessary to analyze the fault propagation process in the system and trace the causes of such dangerous faults.

4.1 Analysis of undetected malicious faults

In this section, the faults that caused a trip signal generation failure and were undetected were analyzed. As shown in the experiment result, 116 malicious faults were undetected.

To identify the root causes for them, first, the source code was analyzed in detail, as shown in Figure 5. The program of the target system is developed based on the block diagrams, and is converted into an assembler code after compiling.

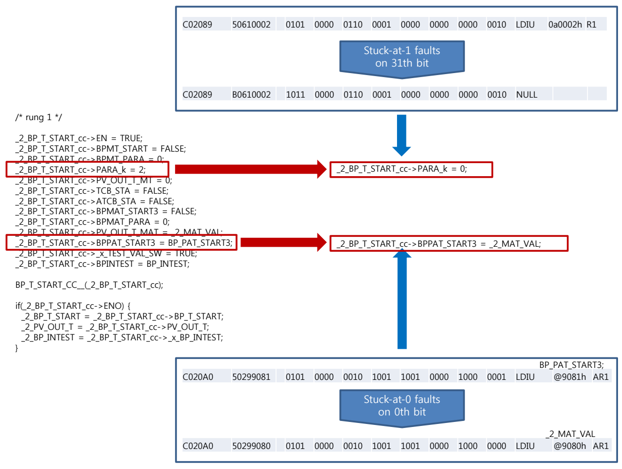


```
/* rung 1 */

_2_BP_T_START_cc->EN = TRUE;
_2_BP_T_START_cc->BPMT_START = FALSE;
_2_BP_T_START_cc->BPMT_PARA = 0;
_2_BP_T_START_cc->PARA_k = 2;
_2_BP_T_START_cc->PV_OUT_T_MT = 0;
_2_BP_T_START_cc->TCB_STA = FALSE;
_2_BP_T_START_cc->ATCB_STA = FALSE;
_2_BP_T_START_cc->BPMAT_START3 = FALSE;
_2_BP_T_START_cc->BPMAT_PARA = 0;
_2_BP_T_START_cc->PV_OUT_T_MAT = _2_MAT_VAL;
_2_BP_T_START_cc->BPPAT_START3 = BP_PAT_START3;
_2_BP_T_START_cc->_x_TEST_VAL_SW = TRUE;
_2_BP_T_START_cc->BPINTEST = BP_INTEST;

BP_T_START_CC__(_2_BP_T_START_cc);

if(_2_BP_T_START_cc->ENO) {
  _2_BP_T_START = _2_BP_T_START_cc->BP_T_START;
  _2_PV_OUT_T = _2_BP_T_START_cc->PV_OUT_T;
  _2_BP_INTEST = _2_BP_T_START_cc->_x_BP_INTEST;
}
```

Fig. 5 Source code and converted assembler.

Figure 6 and Figure 7 show examples for the analysis process to identify the reasons of undetected malicious faults. Followings are two examples of the analysis results of two undetected faults causing no-trip[12].

- Example 1: If a stuck-at-1 fault is injected on the 31st bit at the address 0x00C02089, the HEX code is changed from 50610002 to B061002, as shown in Figure 6. The operator on this address is LDIU, but the fault changes the operator as NULL. In the result, a variable, which is supposed to be 2, becomes 0. Because of this abnormal parameter, the operation mode is stuck at the test mode as shown in Fig. 7, and this causes trip signal generation failure and fault detection failure.
- Example 2: If a stuck-at-0 fault is injected on the 0th bit at the address 0x00C020A0, the HEX code is changed from 50299081 to 50299080, as shown in Figure 6. This fault changes the reference address to @9080h, which should be @9081h. Because of this wrong reference address, the operation mode is stuck at the test mode as shown in Fig. 7, and this causes trip signal generation failure and fault detection failure.



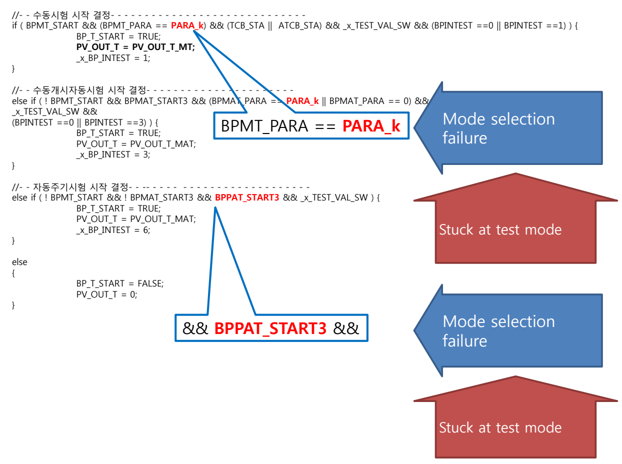
```
C02089   50610002   0101 0000 0110 0001 0000 0000 0000 0010   LDIU   0a0002h R1
C02089   B0610002   1011 0000 0110 0001 0000 0000 0000 0010   NULL
```

```
/* rung 1 */

_2_BP_T_START_cc->EN = TRUE;
_2_BP_T_START_cc->BPMT_START = FALSE;
_2_BP_T_START_cc->BPMT_PARA = 0;
_2_BP_T_START_cc->PARA_k = 2;
_2_BP_T_START_cc->PV_OUT_T_MT = 0;
_2_BP_T_START_cc->TCB_STA = FALSE;
_2_BP_T_START_cc->ATCB_STA = FALSE;
_2_BP_T_START_cc->BPMAT_START3 = FALSE;
_2_BP_T_START_cc->BPMAT_PARA = 0;
_2_BP_T_START_cc->PV_OUT_T_MAT = _2_MAT_VAL;
_2_BP_T_START_cc->BPPAT_START3 = BP_PAT_START3;
_2_BP_T_START_cc->_x_TEST_VAL_SW = TRUE;
_2_BP_T_START_cc->BPINTEST = BP_INTEST;

BP_T_START_CC__(_2_BP_T_START_cc);

if(_2_BP_T_START_cc->ENO) {
  _2_BP_T_START = _2_BP_T_START_cc->BP_T_START;
  _2_PV_OUT_T = _2_BP_T_START_cc->PV_OUT_T;
  _2_BP_INTEST = _2_BP_T_START_cc->_x_BP_INTEST;
}
```

```
_2_BP_T_START_cc->PARA_k = 0;
_2_BP_T_START_cc->BPPAT_START3 = _2_MAT_VAL;
```

```
                                                                       BP_PAT_START3;
C020A0   50299081   0101 0000 0010 1001 1001 0000 1000 0001   LDIU   @9081h  AR1
                                                                       _2_MAT_VAL
C020A0   50299080   0101 0000 0010 1001 1001 0000 1000 0000   LDIU   @9080h  AR1
```

Fig. 6 Fault effect analysis[12].



```
//- - 수동시험 시작 결정- - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
if ( BPMT_START && (BPMT_PARA == PARA_k) && (TCB_STA ||  ATCB_STA) && _x_TEST_VAL_SW && (BPINTEST ==0 || BPINTEST ==1) ) {
                BP_T_START = TRUE;
                PV_OUT_T = PV_OUT_T_MT;
                _x_BP_INTEST = 1;
}

//- - 수동개시자동시험 시작 결정- - - - - - - - - - - - - - - - - - - - -
else if ( ! BPMT_START && BPMAT_START3 && (BPMAT_PARA == PARA_k || BPMAT_PARA == 0) &&
_x_TEST_VAL_SW &&
(BPINTEST ==0 || BPINTEST ==3) ) {
                BP_T_START = TRUE;
                PV_OUT_T = PV_OUT_T_MAT;
                _x_BP_INTEST = 3;
}

//- - 자동주기시험 시작 결정- - -- - - - - -  - - - - - - - - - - - - - - - - - -
else if ( ! BPMT_START && ! BPMAT_START3 && BPPAT_START3 && _x_TEST_VAL_SW ) {
                BP_T_START = TRUE;
                PV_OUT_T = PV_OUT_T_MAT;
                _x_BP_INTEST = 6;
}

else
{
                BP_T_START = FALSE;
                PV_OUT_T = 0;
}
```

Fig. 7 Identification of the reasons of undetected faults[12].

4.2 Fault effect propagation model

As described above, some faults do not generate a trip signal and are not detected. These faults are very significant for the plant safety and must be analyzed and eliminated. However, a manual analysis requires a lot of time and effort for the fault reason identification, and thus the analysis tasks can be effectively performed using a fault effect propagation model.

For the fault effect propagation model development, all variables in the system and their relations should be analyzed. Fig. 8 shows an example of the variable relation definition results.

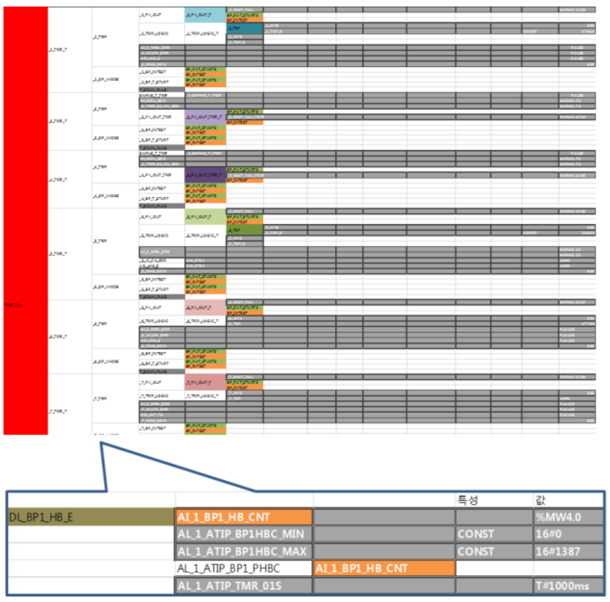

특성
값
DL_BP1_HB_E
AI_1_BP1_HB_CNT
%MW4.0
AL_1_ATIP_BP1HBC_MIN
CONST
16#0
AL_1_ATIP_BP1HBC_MAX
CONST
16#1387
AL_1_ATIP_BP1_PHBC
AI_1_BP1_HB_CNT
AL_1_ATIP_TMR_01S
T#1000ms

Fig. 8 Variables and their relations of the system[12].

Figure 9 shows the developed fault effect propagation model using a fault tree technique. The model was developed with variable level and each basic node represents a variable or register. If a basic node is triggered, then the propagation path is shown in the model as shown in Fig. 6 [12].

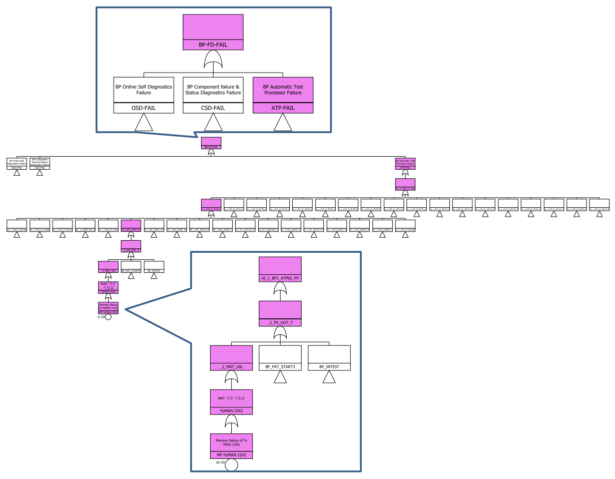

BP-FD-FAIL
BP Online Self Diagnostics Failure
OSD-FAIL
BP Component failure & Status Diagnostics Failure
CSD-FAIL
BP Automatic Test Processor Failure
ATP-FAIL

Fig. 9 A fault propagation model[12].

4 Discussions

This work proposed a method to evaluate fault detection coverage based on fault injection experiments and to identify root causes of the undetected faults. Followings are issues which should be discussed for more reliable results.

- There are limitations of the fault injection experiments into memory. In order to justify the assumption, a fault mapping table which shows the relation between hardware faults and memory faults is necessary. For example, if there is a hardware fault in the input channel 1 of an input card, the variables provided through this channel will be wrong. In this case, injecting faults into the memory area in which these variables are stored can represent the hardware fault. Therefore, the fault in the input channel 1 of an input card can be matched to the fault of memory area assigned to the variables provided through this channel. To develop the fault mapping table, detailed system analysis data such as FMEA (Failure Mode and Effect Analysis) is required.
- While the fault propagation model is developed with variable level, the fault injection experiment is performed with a bit level. Resolutions are not matched in both approaches. If a model is developed with bit level, the number of required basic nodes are too huge to be implemented. For example, in the target system, the size of a variable is 32 bits. If the model is modified with bit level, then the size of model increases 32 times. If the model development and root cause analysis process are automated, then more detailed analysis can be performed.
- Major root causes of the faults which are not detected and causes no-trip are kinds of common cause failure of fault-tolerant techniques and the system. From the analysis results, it was shown that malfunctions, which occur in commonly used functions such as logic or mode selection, lead these common cause failures.

5 Conclusions

In this work, the fault detection coverage of a digital I&C system was evaluated and the undetected faults were analyzed. First, fault injection experiments were performed to evaluate fault detection coverage and to observe the effects of the faults in a system. A software-implemented fault injection technique in which faults can be injected into the memory was used based on the assumption that all faults in a system are reflected in the faults in the memory. In the experiments, the effect of a fault on the system output was observed. Also, a success or failure to detect the fault by fault-tolerant functions included in the system was identified. Second, a fault tree model was developed to identify propagation path from hardware faults to system failures. And the root causes of the undetected faults causing no-trip were identified. Based on the analysis results of the proposed method, it is possible to not only evaluate the system reliability but also identify weak points of fault-tolerant techniques by analyzing undetected faults. The results can be reflected in the designs to improve the capability of fault-tolerant techniques.

Acknowledgement

This work was supported by Nuclear Research & Development Program of the National Research Foundation of Korea grant, funded by the Korean government, Ministry of Science, ICT & Future Planning (Grant Code: 2012M2A8A4025991).

References

[1] LEE, S. J., *et al.*: Reliability assessment method for NPP digital I&C systems considering the effect of automatic periodic tests, Annals of Nuclear Energy, 2010, 37: 1527-1533.

[2] KANG, H. G., *et al.*: An overview of risk quantification issues of digitalized nuclear power plants using static fault tree, Nuclear Engineering and Technology, 2009, 41: 849-858.

[3] KANG, H. G., and SUNG, T.: A quantitative study on important factors of the PSA of safety-critical digital systems, Nuclear Engineering and Technology, 2001, 33: 596-604.

[4] KANG, H. G., and SUNG, T.: An analysis of safety-critical digital systems for risk-informed design, Reliability Engineering and System Safety, 2002, 78:. 307-314.

[5] KWON, K. C., LEE, and LEE M. S..: Technical Review on the Localized Digital Instrumentation and Control Systems, Nuclear Engineering and Technology, 2009, 41: 447-454.

[6] PARK, J. H.. LEE, D. Y., and KIM, C. H.: Development of KNICS RPS prototype. Proceeding of ISOFIC-2005, 2005.

[7] CHOI, J. G., *et al.*: Fault Detection Coverage Quantification of Automatic Test Functions of Digital I&C System in NPPs, Nuclear Engineering and Technology, 2012, 44: 421-428.

[8] PINNA, T., BOCCACCINI, L. V., and SALAVYV, J. F.: Failure mode and effect analysis for the European test blanket modules, Reliability Engineering and System Safety, 2008, 83: 1733-1737.

[9] HSUEH, M., TSAI, T. K., and IYER, R. K.: Fault injection techniques and tools, IEEE Computer, 1997, 30: 75-82.

[10] LEE, S. J., and JUNG, W.: Experimental Approach to Evaluate the Reliability of Digital I&C Systems in Nuclear Power Plants, Proceeding of PSAM12, 2014.

[11] Texas Instruments, 1994. Code Composer, User's Guide.

[12] LEE, S. J., and JUNG, W.: Effect Analysis of Digital I&C Systems on Plant Safety based on Fault-Tree Analysis, Proceeding of KNS Spring Meeting, 2014.