

Automated Exhaustive Test Case Generation for FBD Software



KNS 2021 Fall Meeting, Oct. 2021

Sang Hun Lee¹, Sung Min Shin¹, Hyun Gook Kang², Jong Gyun Choi¹

¹Korea Atomic Energy Research Institute

²Rensselaer Polytechnic Institute

Contents

- **Introduction**

- Research Background & Objectives

- **Method**

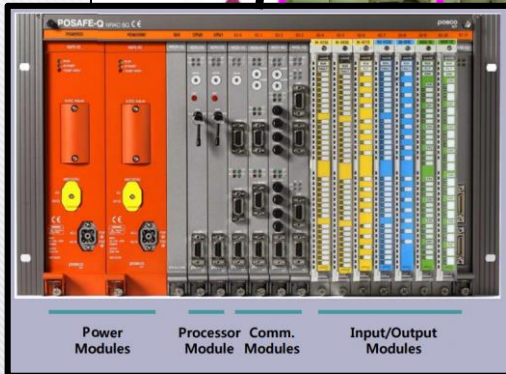
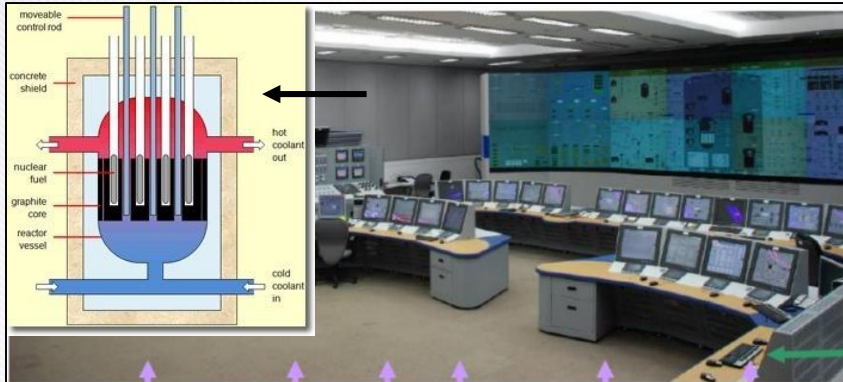
- Translation of FBD to SMT formula
- Exhaustive test case generation using SMT Solver

- **Application**

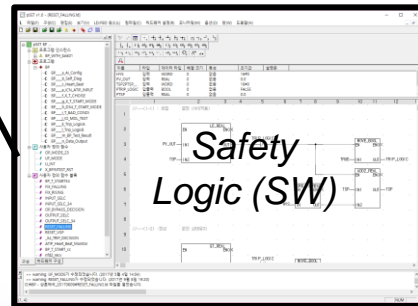
- RPS Trip Logic Software Testing

- **Conclusion**

Software is important for NPP Safety



PLC Modules (HW)

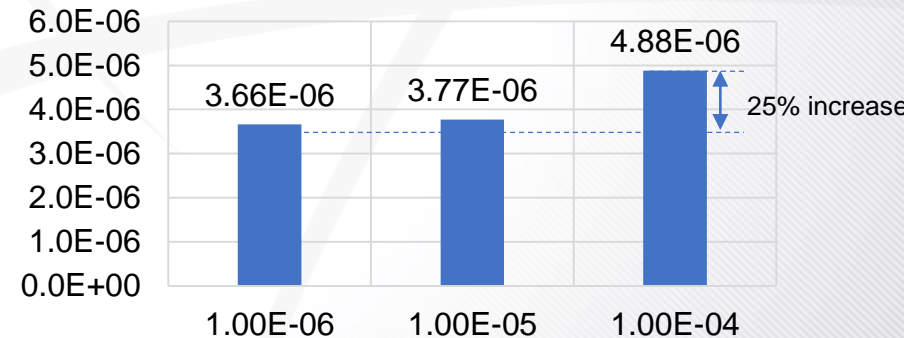


PLC used in NPP safety I&C systems

Digital system failure events reported in LERs

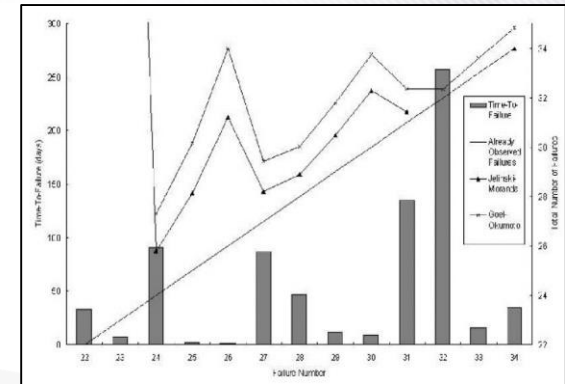
Category	Number of Event
Software error	30 (39.97%)
HMI error	25 (31.65%)
EM interference	15 (18.99%)
Random HW failure	9 (11.39%)
Total	79

RPS unavailability for various software failure probability

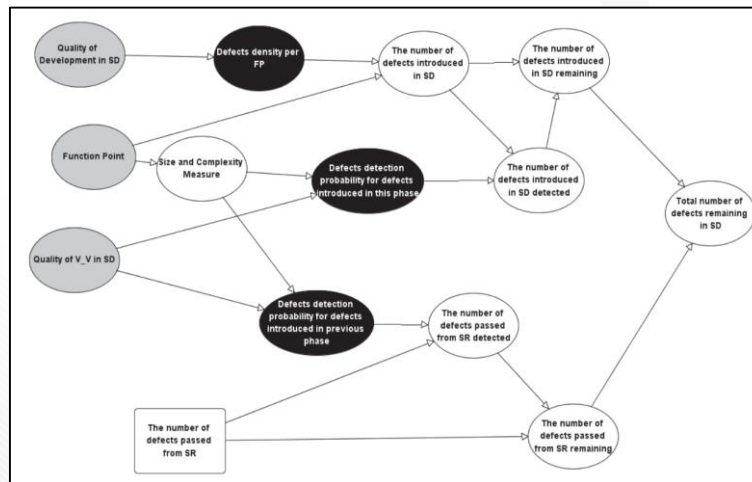


Quantitative SW Reliability Assessment

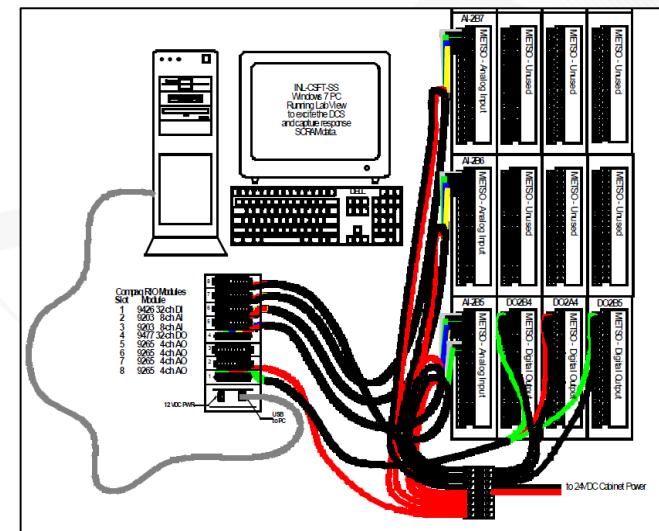
- Software Reliability Growth Model
- Bayesian Belief Network Model
- Metric-based Method
- Context-based Software Reliability Method
- Test-based Method
- Etc...



SRGM application to NPP SW



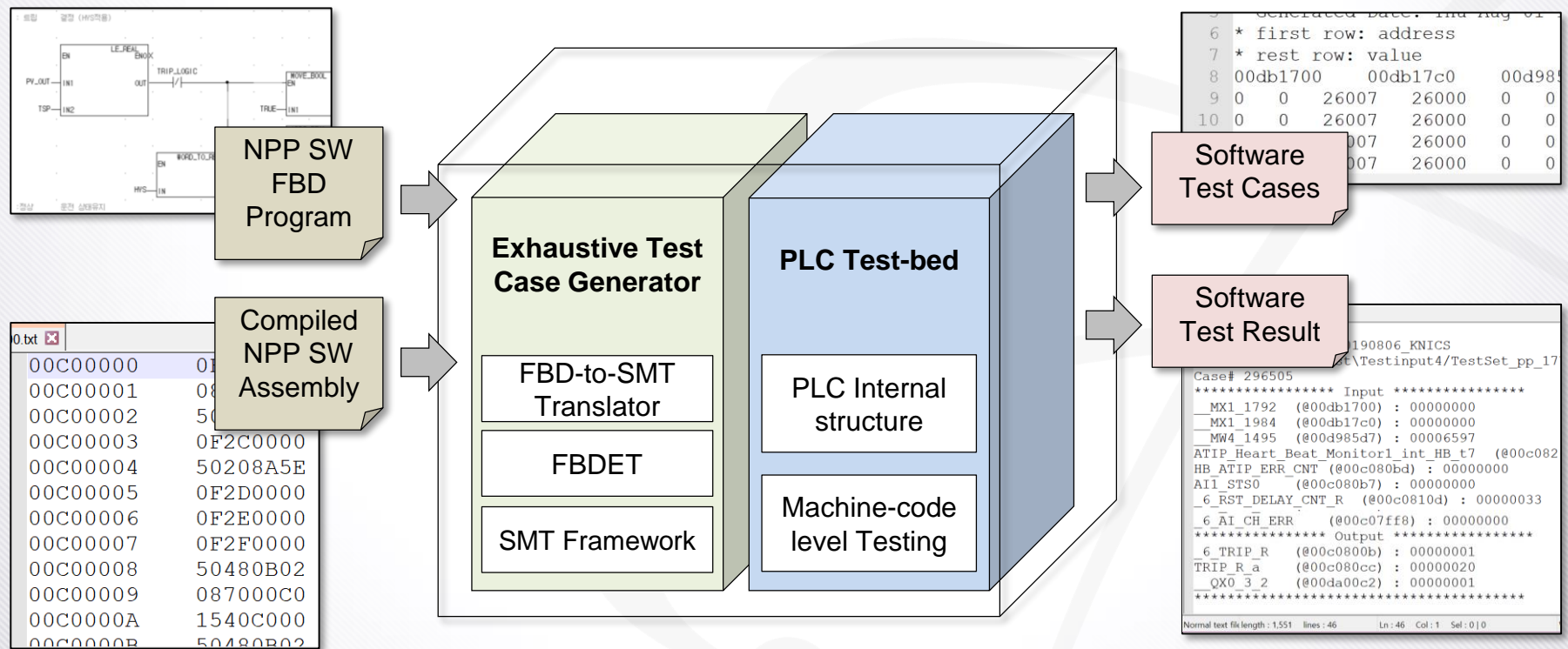
BBN model for NPP SW



Test Environment for ATR LOCS SW

Research Objectives

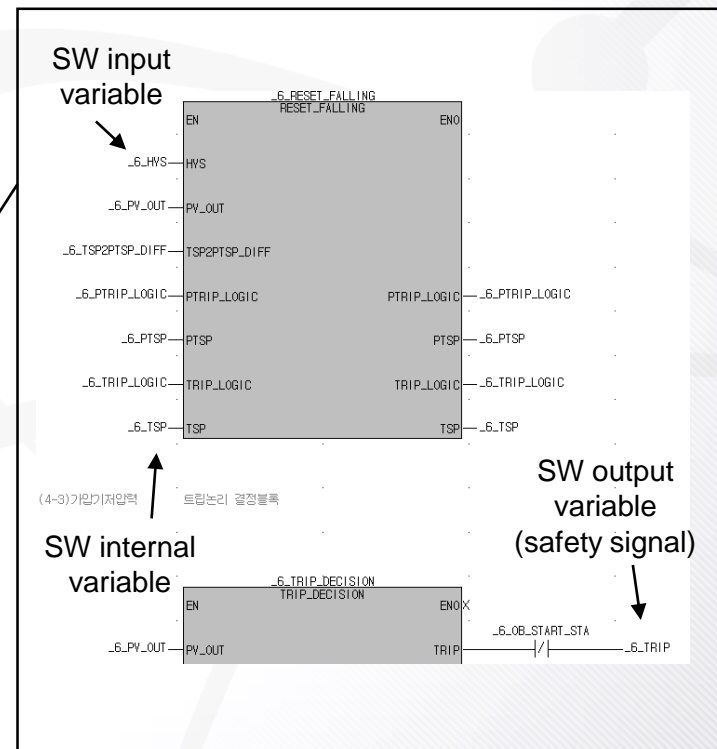
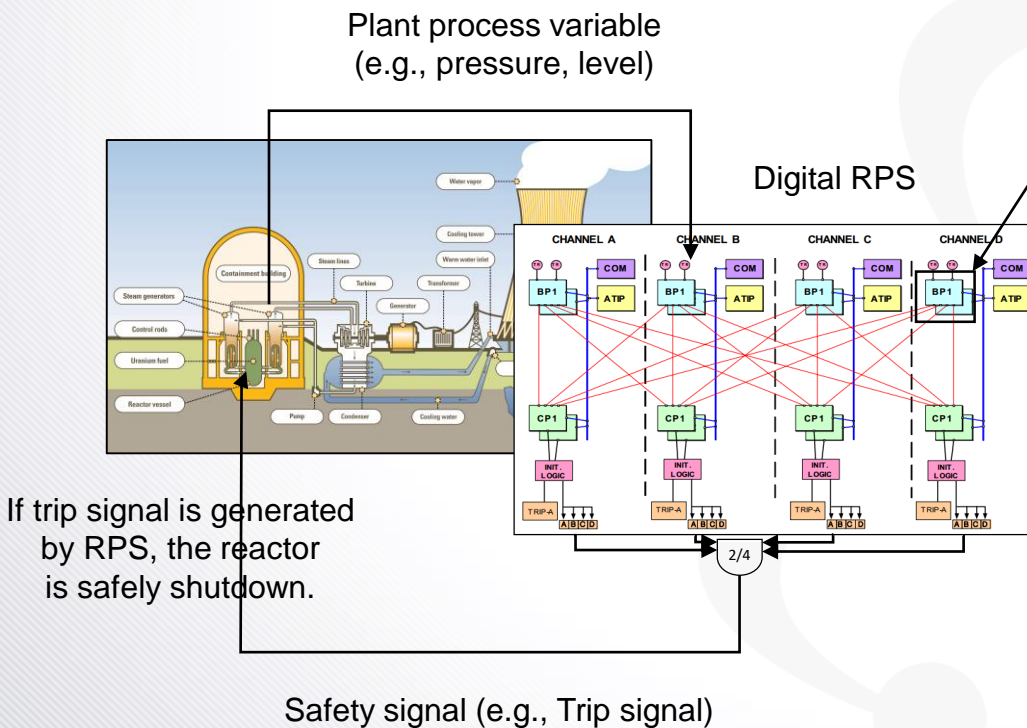
- Integrated Testing Framework for NPP Safety Software
 - Simulation-based Test-bed for PLC Software Testing
 - **SMT-based Exhaustive Test Case Generation for PLC Software**



Block Diagram of NPP Software Exhaustive Testing Framework

Definition of Exhaustive Test Case

- **Software failure**: fail to generate safety signal when demand comes.
 - Demand: a plant condition that requires actuation of safety systems
- **Exhaustive test case** : all possible sets of the SW input/internal variables that generates SW safety output (e.g. trip signal).



Example of Reactor Trip Logic Software

Theory of Test Case Generation

- **Generating Test Cases → Satisfiability Problem**

- Satisfiability : A formula is satisfiable if there is a model that makes formula true.

- **Examples of Satisfiability Problems**

- Boolean Satisfiability Problem (SAT) : Propositional formula
- Satisfiability Modulo theory (SMT) : First order logic formula

$$f_1 = x_1 \wedge x_2 \qquad f_2 = x_1 \wedge \neg x_1$$

$$f_3 = (\neg x_1 \vee x_2) \wedge (\neg x_1 \vee x_3) \wedge (x_2 \vee x_3) \wedge (\neg x_2 \vee \neg x_3)$$

$$f_4 = (x_1 + 5 = x_2) \wedge (x_2 \leq x_3) \wedge (x_3 \leq 8)$$

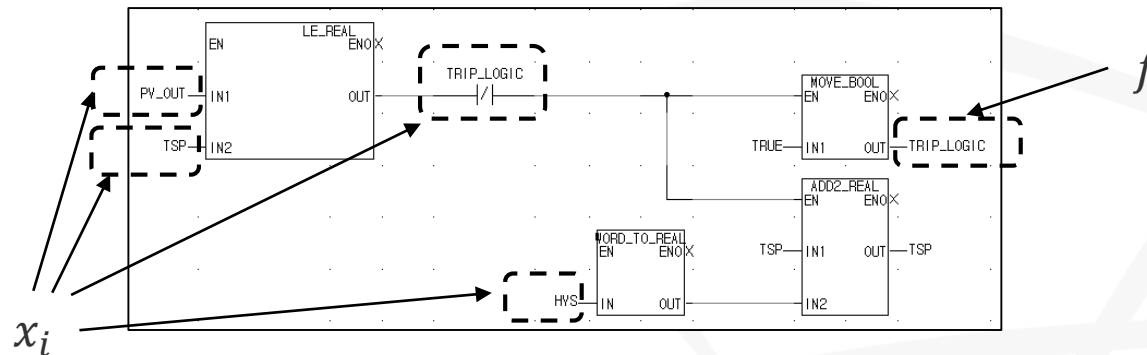
- Is the formula (f_1, f_2, f_3, f_4) satisfiable?
- If the formula is satisfiable, do we have x_i 's that make formula true?

Theory of Test Case Generation

- **Formulation of Test Case Generation Problem**

- **Objective** : Derive all possible software input/internal variables (x_i 's) such that the software output ($f(x_i)$) becomes true.

$$\underset{x_i \in S}{\mathit{arg}} f(x_i) = \mathit{true}$$

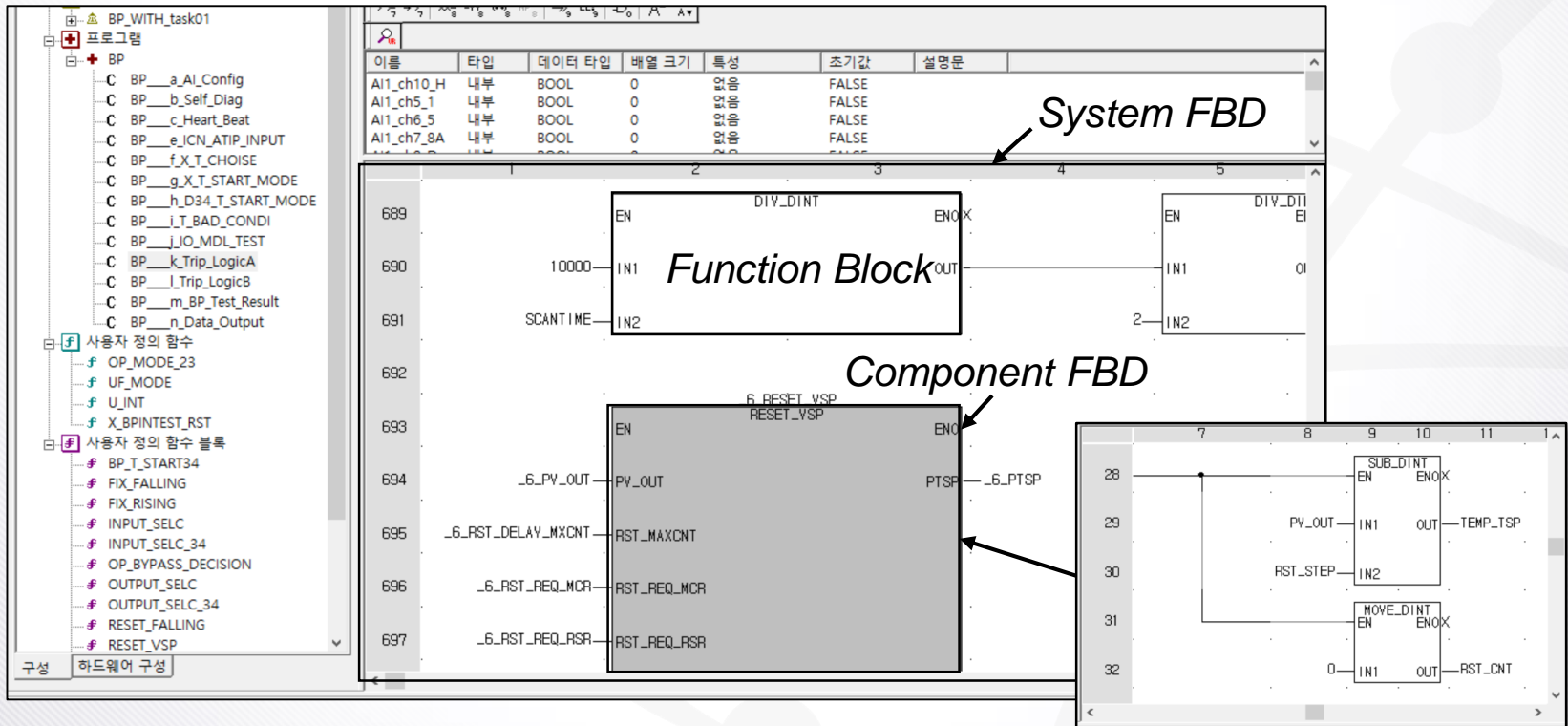


An example FBD program for calculating `TRIP_LOGIC`

- **Formula (f)** : Software output
- **Literals (x_i)** : Software input / internal variables
- **Space (S)** : Domain of all possible software input / internal variables that may encounter during software operation

NPP safety software - FBD program

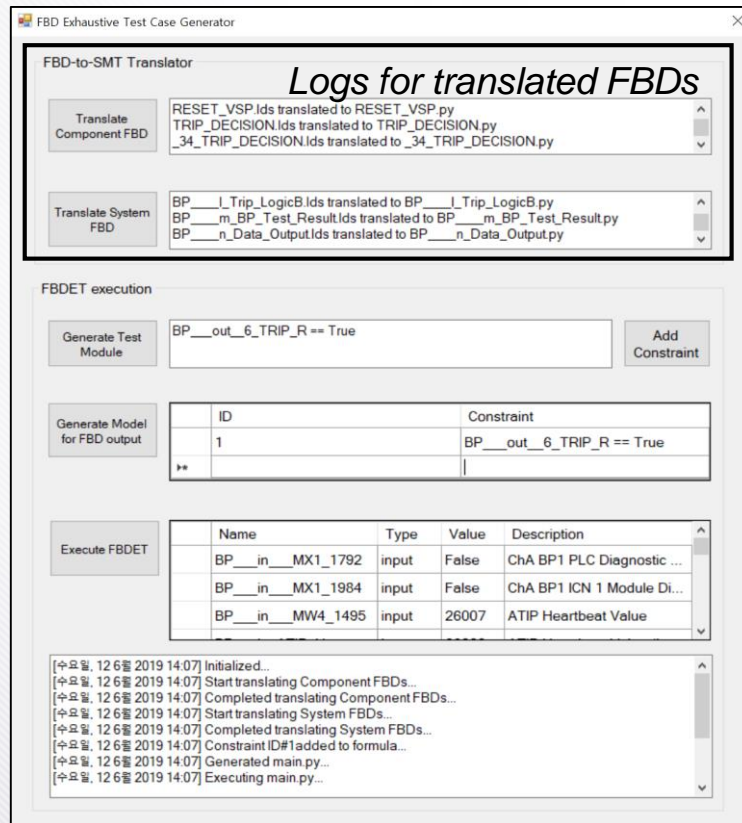
- **Function Block** : Basic unit having input/internal/output variables
- **Component FBD** : Group of FBs interconnected according to execution order
- **System FBD** : Group of component FBDs (a whole software)



An example of FBD program

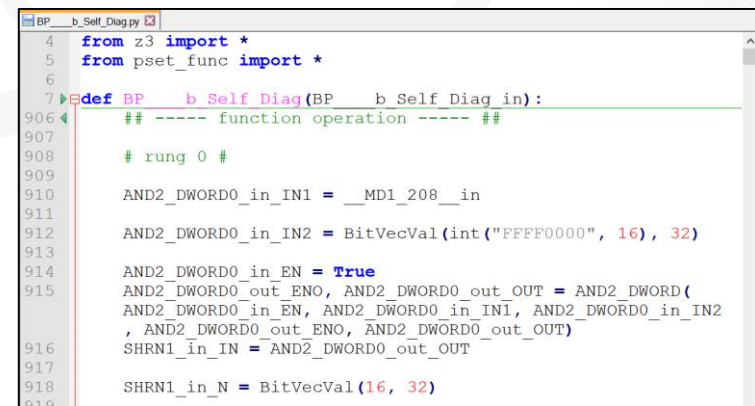
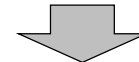
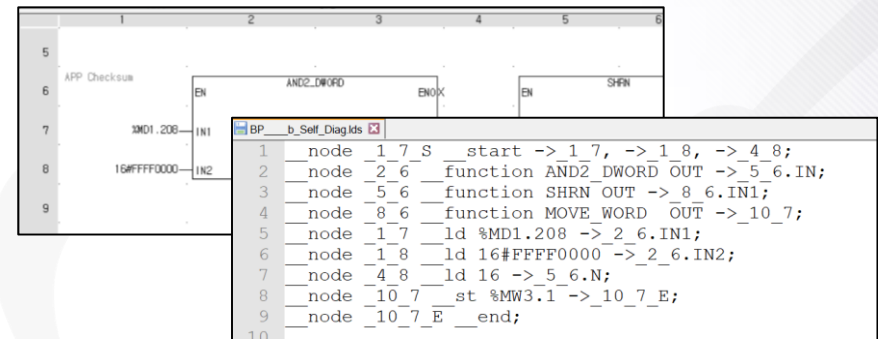
Translation of FBD programs

- FBD exhaustive test case generator (FETCG) : automatically translates the source code to SMT formula based on the developed translation rules.



Screenshot of exhaustive test case generator

Source code of PLC program

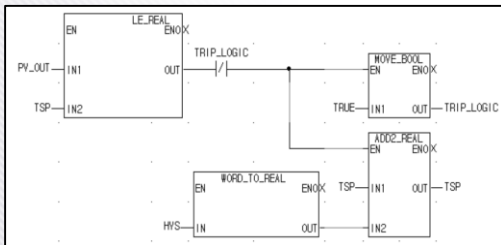


Translated SMT formula for the example program

Single Test Case Generation

Translated FBD program

```
RESET_FALLING.py
4 from z3 import *
5 from pset_func import *
6
7 def RESET_FALLING(EN, in_HYS, in_PV_OUT, i
8
9     ## ----- temporary variables ----- ##
10
11     temp1 = False
12     temp2 = False
13     ENO_1 RESET_FALLING = False
14
15     ## ----- function operation ----- ##
16
17     # rung 0 #
18
19     LE_REAL0_IN1 = in_PV_OUT
20
21     LE_REAL0_IN2 = in_TSP
22
23     LE_REAL0_EN = True
24     ENO_1 RESET_FALLING, OUT_1 RESET F
25     LE_REAL0_OUT = OUT_1_RESET_FALLING
```



1) Define Program under Test

1-1. Define variables

1-2. Define program

2) Define Test Requirements

2-1. Define desired software output
(e.g., *TRIP_LOGIC* = true)

3) Solve for Test Requirements

3-1. Check Satisfiability (→ sat/unsat)

3-2. Retrieve model for the
program variables (solution)

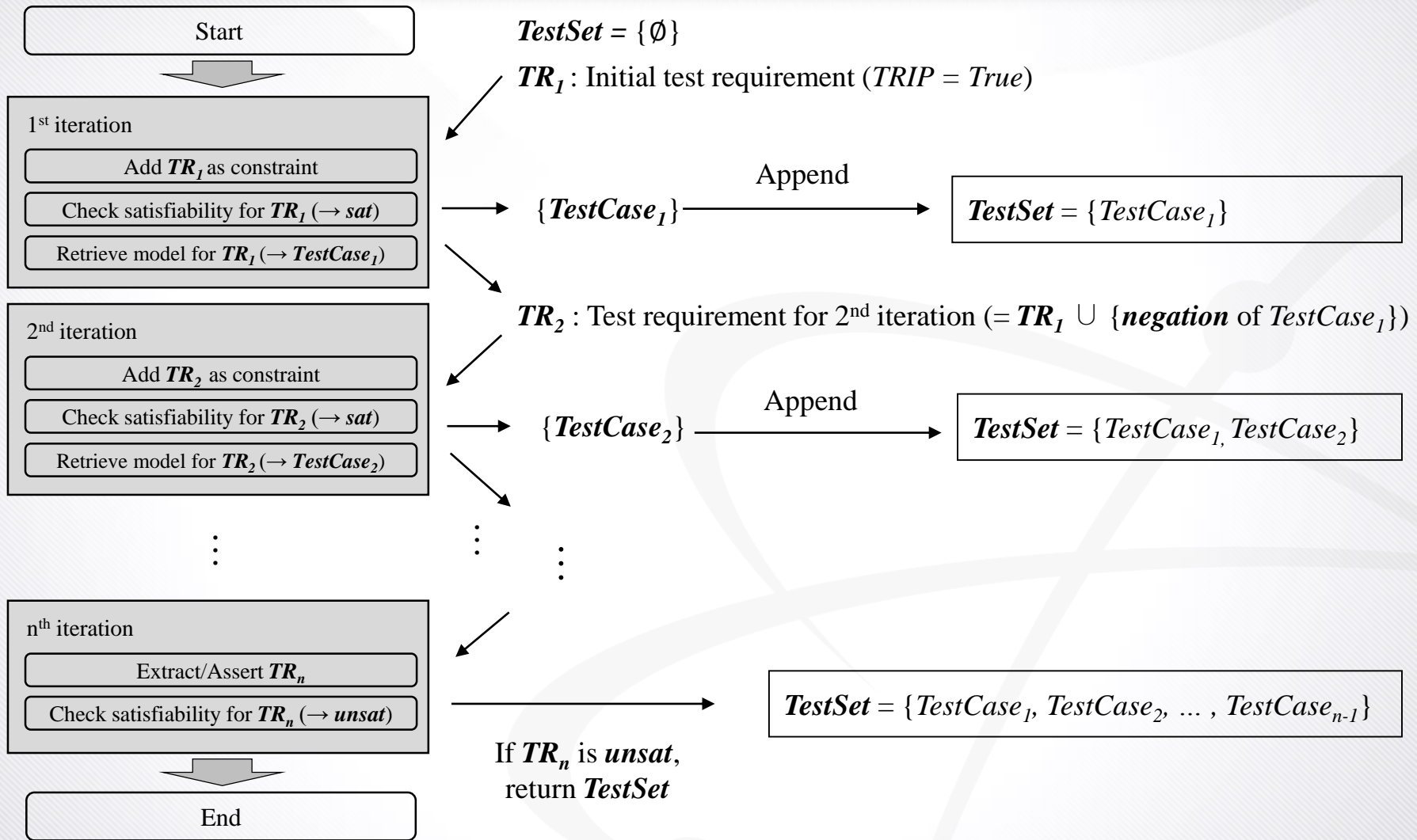
```
cmd
C:\#z3-master\examples\python>python main.py
* checking the model satisfiability ...
sat
* retrieving an interpretation ...
[PV_OUT = 17780, TSP = 17780, TRIP_LOGIC = False]
```



Generated test case

Process of single test case generation for PLC program

Exhaustive Test Case Generation Algorithm



Process of test case generation for FBD program

Exhaustive Test Case Generation Algorithm

- The algorithm is implemented to FBD exhaustive test case generator.
- It automatically generates exhaustive test cases for PLC program given the defined software output.

FBD Exhaustive Test Case Generator

FBD-to-SMT Translator

Translate Component FBD
 RESET_VSP.lids translated to RESET_VSP.py
 TRIP_DECISION.lids translated to TRIP_DECISION.py
 _34_TRIP_DECISION.lids translated to _34_TRIP_DECISION.py

Translate System FBD
 BP__l_Trip_LogicB.lids translated to BP__l_Trip_LogicB.py
 BP__m_BP_Test_Result.lids translated to BP__m_BP_Test_Result.py
 BP__n_Data_Output.lids translated to BP__n_Data_Output.py

FBDET execution

(1) Define desired software output

Generate Test Module
 BP__out_6_TRIP_R == True

Generate Model for FBD output

ID	Constraint
1	BP__out_6_TRIP_R == True

(2) Define range/value of software variables

Name	Type	Value	Description
BP__in__MX1_1792	input	False	ChA BP1 PLC Diagnostic ...
BP__in__MX1_1984	input	False	ChA BP1 ICN 1 Module Di...
BP__in__MW4_1495	input	26007	ATIP Heartbeat Value

(3) Program log

```
[중요] 12 6월 2019 14:07 Initialized...
[중요] 12 6월 2019 14:07 Start translating Component FBDs...
[중요] 12 6월 2019 14:07 Completed translating Component FBDs...
[중요] 12 6월 2019 14:07 Start translating System FBDs...
[중요] 12 6월 2019 14:07 Completed translating System FBDs...
[중요] 12 6월 2019 14:07 Constraint ID#1 added to formula...
[중요] 12 6월 2019 14:07 Generated main py...
[중요] 12 6월 2019 14:07 Executing main.py...
```

Screenshot of exhaustive test case generator

Source code of program

```

1 node 1_7 S_start -> 1_7, -> 1_8, -> 4_8;
2 node 2_6 function AND2_DWORD OUT -> 5_6.IN;
3 node 5_6 function SHR_N OUT -> 8_6.IN1;
4 node 8_6 function MOVE_WORD OUT -> 10_7;
5 node 1_7 ld %MD1.208 -> 2_6.IN1;
6 node 1_8 ld 16#FFFF0000 -> 2_6.IN2;
7 node 4_8 ld 16 -> 5_6.N;
8 node 10_7 st %MW3.1 -> 10_7_E;
9 node 10_7 E_end;

```

```

6 * first row: address
7 * rest row: value
8 00db1700 00db17c0 00d985d7 00c0822e 00c080bd 00c080b7
00c0810d 00db46b0 00c080b8 00d985fc 00c080cc 00c080cd
00c080c5 00c080c6 00c07f93 00c07f94 00db3049 00db304a
00db304b 00c08275 00db4663 00c08276 00c07fad 00c08009
00c08105 00c08115 00c07ffb 00c089e8 00c089e9 00c089e6
00c089e7 00c07f8c 00c07ffb
9 0 0 26007 26000 0 0 51 10 0 0 0 0 32 0 0 0
0 0 0 0 0 0 1 0 17790 17790 0 0 0 0 0 0
10 0 0 26007 26000 0 0 51 10 0 0 0 0 32 0 0 0
0 0 0 0 0 0 1 0 17787 17790 0 0 0 0 0 0
11 0 0 26007 26000 0 0 51 10 0 0 0 0 32 0 0 0
0 0 0 0 0 0 1 0 17786 17790 0 0 0 0 0 0
12 0 0 26007 26000 0 0 51 10 0 0 0 0 32 0 0 0
0 0 0 0 0 0 1 0 17788 17790 0 0 0 0 0 0

```

Generated test case file for Rx. trip signal output

Demonstration of Test Case Generator

FBD Exhaustive Test Case Generator (FETCG)

FBD-to-SMT Translator

Translate Component FBD

Translate System FBD

FBD-ET execution

Generate Test Module Add Constraint

ID	Constraint
*	

Load Variable Information

Name	Type	Value	Description	Address
*				

Execute FBD-ET

[Wednesday, 16 October 2019 12:09] Initialized...

pSET v1.0 - [RESET_FALLING.ld]

FBD 원소(L) 컴파일(C) 하드웨어 설정(R) 모니터링(M) 옵션(O) 창(W) 도움말(H)

이름	타입	데이터 타입	배열 크기	특성	초기값	설명문
HYS	입력	WORD	0	정수	16#0	
PV_OUT	입력	REAL	0	실수	0.0	
TSP2TSP...	입력	WORD	0	정수	16#0	
PTRIP_LOGIC	입출력	BOOL	0	진/거짓	FALSE	
PTSP	입출력	REAL	0	실수	0.0	
TRIP_LOGIC	입출력	BOOL	0	진/거짓	FALSE	
TSP	입출력	REAL	0	실수	0.0	

1 //---(1-1) : 트립 결정 (HYS적용)

2

3

4

5

6

7

8 //---(1-2) : 정상 운전 상태유지


9

NUM

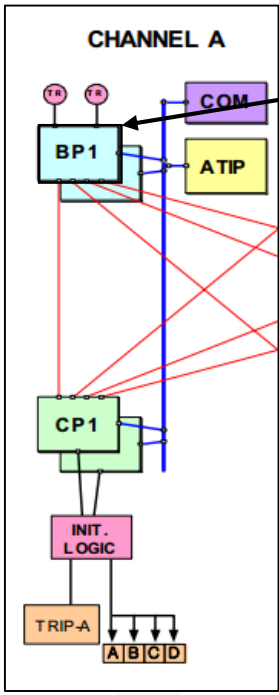
(8, 1)

Application: RPS Trip Logic Software

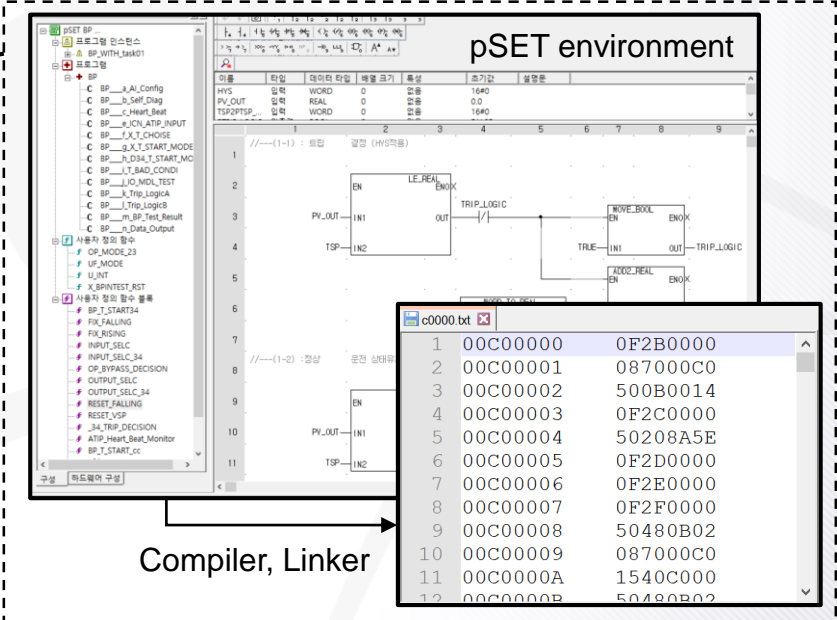
- IDiPS-RPS was developed in Korea to satisfy the design requirements of APR-1400 which has full-digital I&C systems.
- BP software has 15 trip logic modules which generate Rx. trip signal when the process parameter deviates from permissible operating condition.



The image shows the physical hardware for the IDiPS-RPS system. On the left, a rack of three green cabinets contains various modules. On the right, a close-up of a rack-mounted unit shows a complex array of cables and connectors.



CHANNEL A block diagram showing the flow of data and control signals. It includes modules for BP1, CP1, INIT. LOGIC, TRIP-A, and ATIP, connected to a COM bus. The diagram shows how process parameters (T_A, T_K) are processed through BP1 and CP1, then through INIT. LOGIC to generate TRIP-A signals (A, B, C, D).



pSET environment showing source code and compiled machine code. The source code is in a ladder logic format, and the compiled machine code is shown in a table.

Address	Machine Code	Hex Code
1	00C00000	0F2B0000
2	00C00001	087000C0
3	00C00002	500B0014
4	00C00003	0F2C0000
5	00C00004	50208A5E
6	00C00005	0F2D0000
7	00C00006	0F2E0000
8	00C00007	0F2F0000
9	00C00008	50480B02
10	00C00009	087000C0
11	00C0000A	1540C000
12	00C0000B	50480B02

Compiler, Linker

Architecture of one channel IDiPS-RPS and prototype

Source code and compiled machine code of BP trip logic software

Application: RPS Trip Logic Software

- A total of 4,206,164,480 test cases were generated for the target BP trip logic and tested in the developed test-bed.
- As a result, all test cases generated the trip output after BP software is executed. (~ 6.205 msec/case)

ID of test case

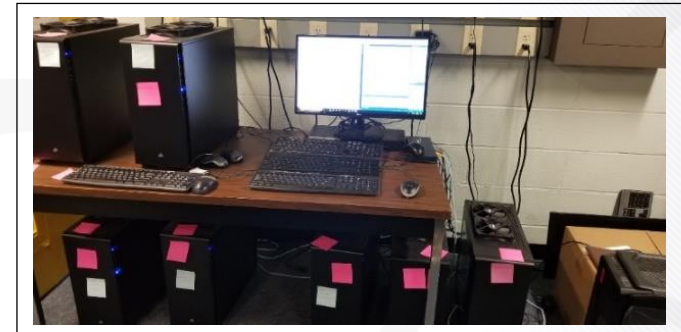
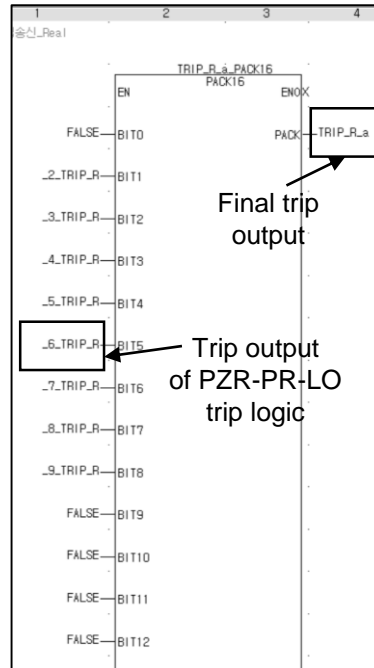
```

File : C:\c_test_20190806\KNICS
(dissertation)\Test\Testinput4\TestSet_pp_1\
Case# 296505

***** Input *****
MX1_1792 (@00db1700) : 00000000
MX1_1984 (@00db17c0) : 00000000
MW4_1495 (@00d985d7) : 00006597
ATIP_Heart_Beat_Monitor1_int_HB_t7 (@00c082
HB_ATIP_ERR_CNT (@00c080bd) : 00000000
AI1_STS0 (@00c080b7) : 00000000
6_RST_DELAY_CNT_R (@00c08104) : 00000033
MW1_1712 (@00db46b0) : 0000000A
AI1_STS1 (@00c080b8) : 00000000
MW4_1532 (@00d985fc) : 00000000
TRIP_R_a (@00c080cc) : 00000000
TRIP_R_b (@00c080cd) : 00000000
PTRIP_R_a (@00c080c5) : 00000020
PTRIP_R_b (@00c080c6) : 00000000
AI_2_MDL_ERR (@00c07f93) : 00000000
AI_3_MDL_ERR (@00c07f94) : 00000000
MW2_73 (@00db3049) : 00000000
MW2_75 (@00db304a) : 00000000
MW2_74 (@00db304b) : 00000000
TON4_int_Start (@00c08275) : 00000000
clk (@00db4663) : 00000000
TON4_int_Residue (@00c08276) : 00000000
T_SCAN_FLAG (@00c07fad) : 00000001
6_TRIP_LOGIC_R (@00c08009) : 00000000
6_PV_OUT_AI (@00c08105) : 0000457e
6_TSP_R (@00c08115) : 0000457e
6_OB_PERM (@00c07ffb) : 00000001
6_RST_REQ_MCR_DI (@00c089e8) : 00000001
6_RST_REQ_RSR_DI (@00c089e9) : 00000000
6_OB_REQ_MCR_DI (@00c089e6) : 00000000
6_OB_REQ_RSR_DI (@00c089e7) : 00000000
AI2_ch6_6 (@00c07f8c) : 00000001
6_AI_CH_BNK (@00c07f80) : 00000000
***** Output *****
6_TRIP_R (@00c0800b) : 00000001
TRIP_R_a (@00c080cc) : 00000020
QX0_3_2 (@00da00c2) : 00000001
    
```

Status of input/
internal
variable
before test
execution in
test-bed

Status of output
variable after
test execution
by test-bed



Test Setup (3.60GHz 640 CPUs)

```

Current working dir: D:\Cpptest\c_test_20170824_Win\Test
0: D:\Cpptest\c_test_20170824_Win\Test\AAA2\ir

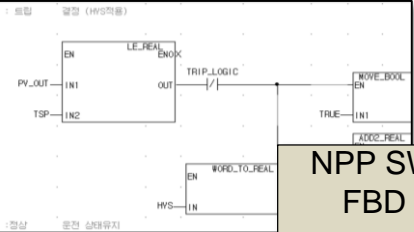
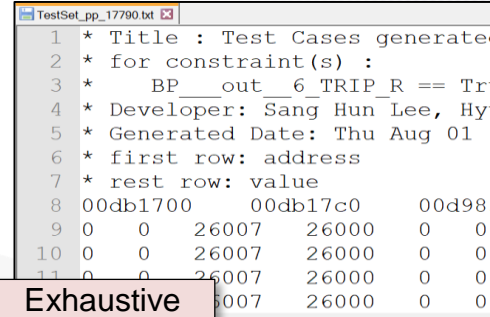
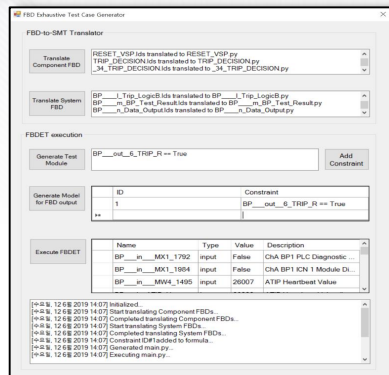
loop time : 0.003
Total # of executed instructions : 98755

No. Opcode Count Time(clock) Count(%) Time(%)
2 ABSI 14 177 0.01 % 0.03 %
6 ADDF 4 1733 0.00 % 0.09 %
7 ADDF3 21 1264 0.02 % 0.34 %
159 LDFGE 30 56 0.03 % 0.02 %
165 LDI 14666 39 14.85 % 7.32 %
166 LDI_LDI 16 78 0.02 % 0.02 %
167 LDI_STI 510 41 0.52 % 0.27 %
168 LDIU 44731 93 45.29 % 52.80 %
173 LDIF0 656 32 0.66 % 0.27 %
    
```

BP software execution in test-bed

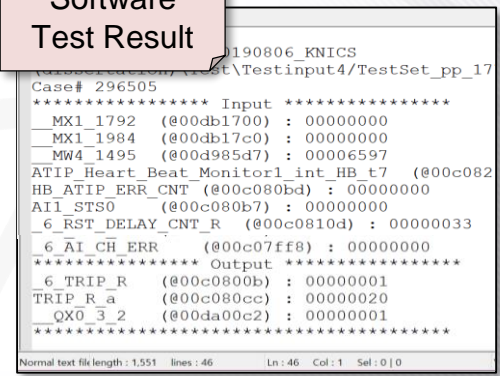
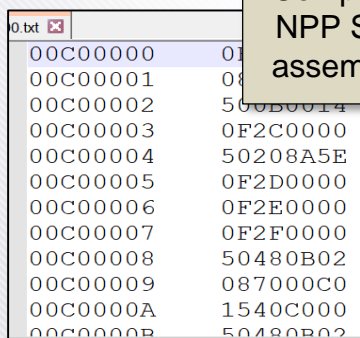
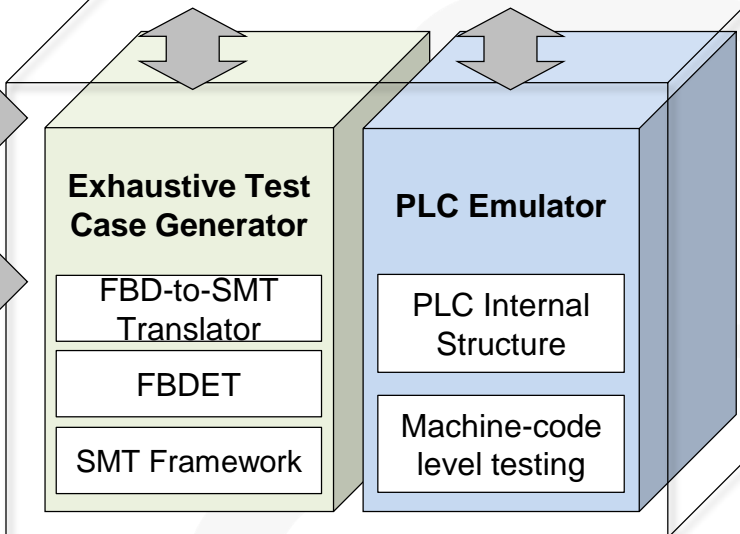
A part of test result for BP target trip logic

Conclusion



NPP SW FBD program

Compiled NPP SW assembly



Rensselaer

UNIST
ULSAN NATIONAL INSTITUTE OF SCIENCE AND TECHNOLOGY



한국원자력연구원
Korea Atomic Energy Research Institute





THANK YOU