FBD_FTA : An Automatic Assistant for Fault Tree Analysis of Function Block Diagrams

Youngju Seo*, Dong-Ah Lee, and Junbeom Yoo

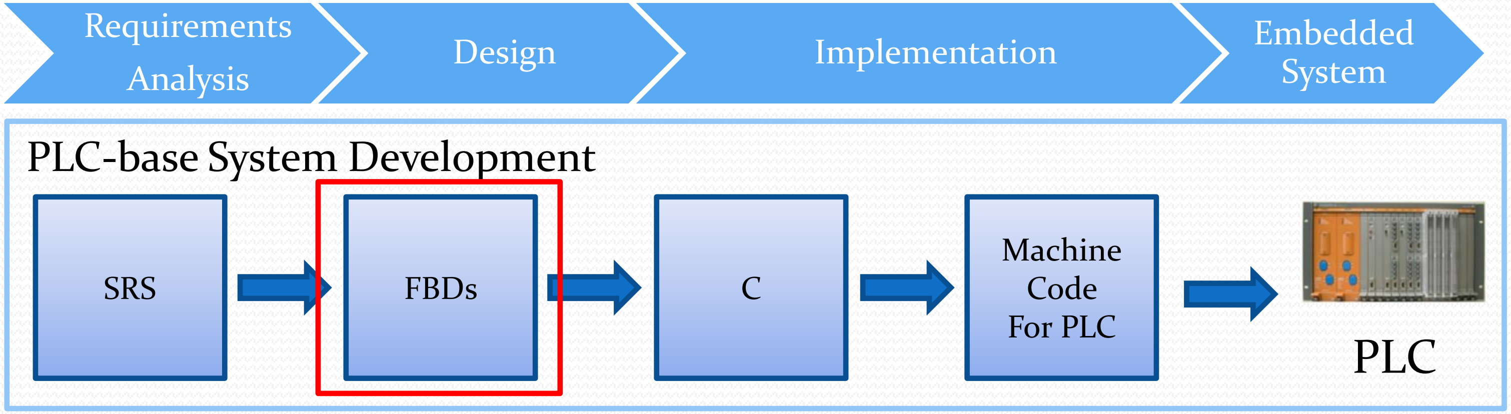
Contents

- Introduction
- Background
- Fault Tree Templates
- FBD_FTA
- Case study
- Conclusion and Future work
- Reference

Introduction

Introduction

- Failures of safety-critical systems
  - Use hazard analysis techniques to assess the hazards.
- Hazard analysis to our target system.
  - It use Programmable logical controllers (PLC).
  - PLC program uses function block diagram (FBD) for programming language.
- FBD_FTA for FTA about FBD program
  - FBD_FTA uses fault tree analysis (FTA) and temporal fault tree (TFT).



Requirements Analysis

Design

Implementation

Embedded System

PLC-base System Development

SRS

FBDs

C

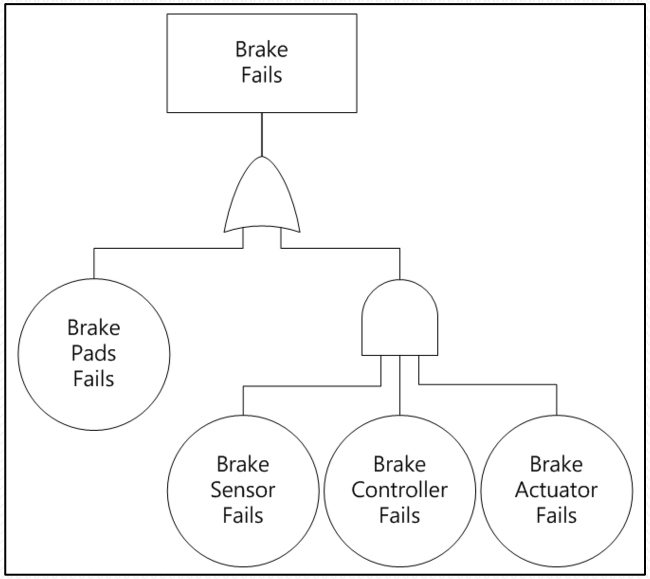
Machine Code For PLC

PLC

Background

Background – Fault tree analysis

- Fault tree analysis (FTA)
  - Provides a method for determining causes of the accident.
  - Draw fault in a square, basic events in circles.
  - Logical gates AND and OR, make relation between events.
  - FT can be written as a boolean expression.



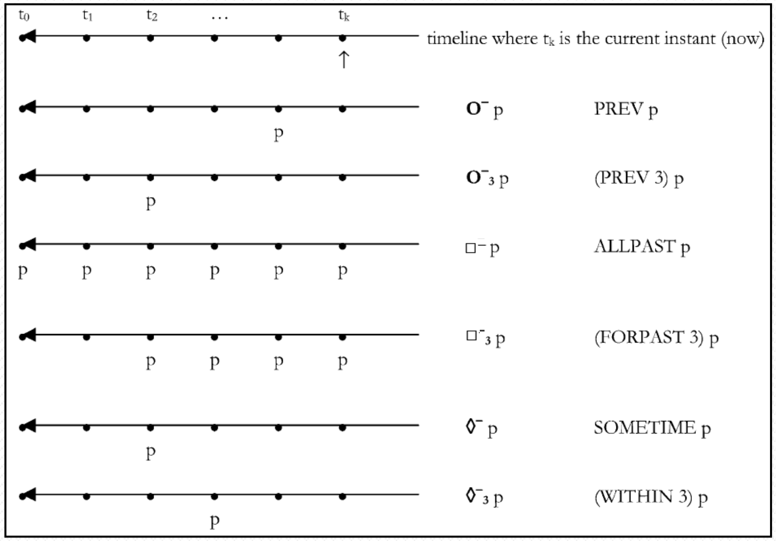
Fault tree

- Fault : Brake Fails
- Relation : OR, AND
- Leaf events : Brake Pads Fails, Brake Sensor Fails, Brake Controller Fails, Brake Actuator Fails
- Boolean expression

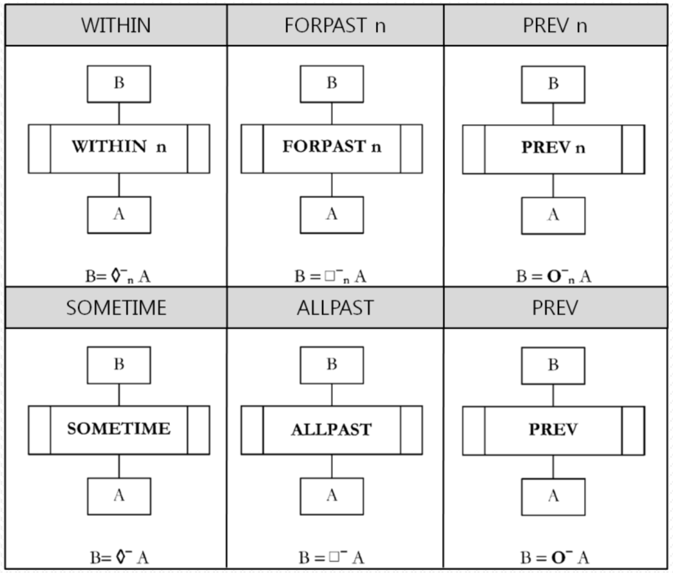
$$\mathit{Brake\ Fails} \rightarrow (\ \mathit{Brake\ pads\ Fails} \lor (\ \mathit{Brake\ Sensor\ Fails} \land \mathit{Brake\ Controller\ Fails} \land \mathit{Brake\ Actuator\ Fails}\ )\ )$$

Background – Temporal fault tree

- Temporal fault tree (TFT)
  - The traditional FT does not handle temporal properties.
  - Temporal fault tree (TFT) uses propositional linear temporal logic(past) (PLTLP) to add temporal properties.
  - The PLTLP uses logical operators and temporal operators.



Example state sequences of some temporal connectives



Temporal gates for fault tree

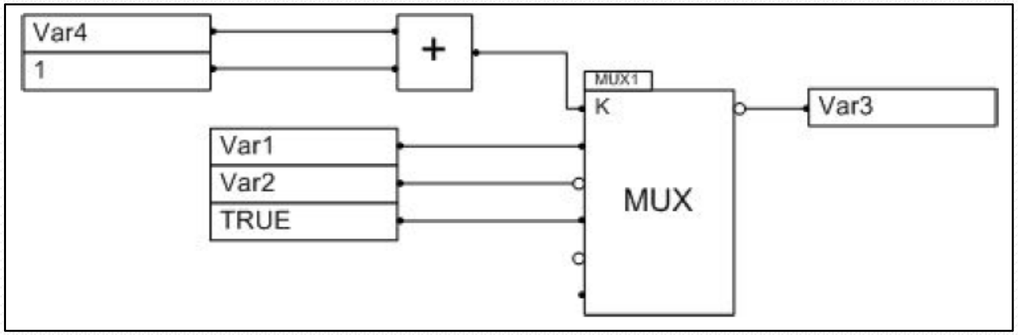
Background – Temporal fault tree

- PLTLP also can be written as a boolean expression and temporal operators.
  - WITHIN n : $\Diamond_n^-$
  - FORPAST : $\Box_n^-$
  - PREV n : O_n^-
  - SOMETIME : $\Diamond^-$
  - ALLPAST : $\Box^-$
  - PREV : O^-
  - TFT expression

1. $rain \rightarrow (\Diamond_2^- (cold \wedge humid))$
2. $lift\_arrives \rightarrow (\Diamond^- (req\_from\_floor \vee req\_from\_passenger))$

Background – Function block diagram

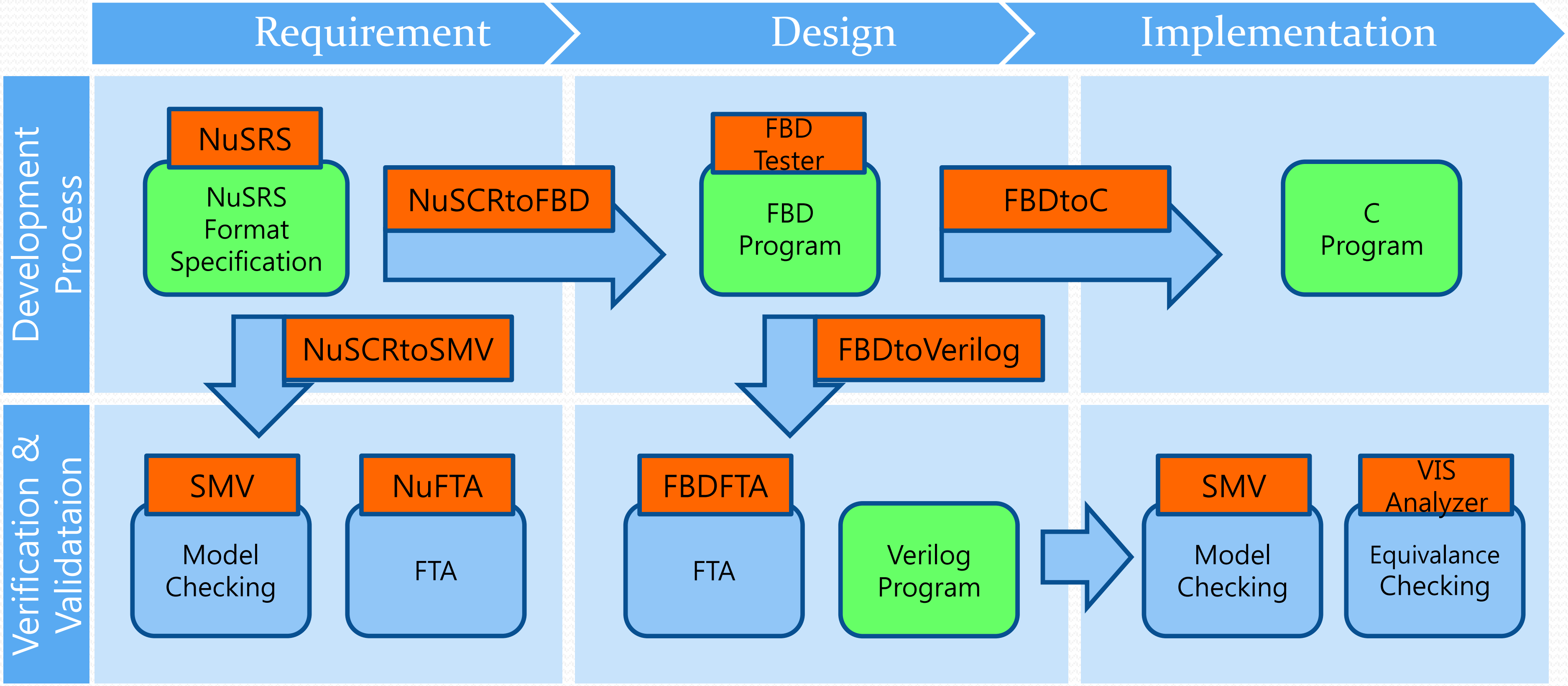
- Function block diagram (FBD)
  - One of programming languages of PLCs.
  - PLCs are used to control of machinery on factory, and nuclear power plant.
  - FBD can describe the function between input variables and output variables.
  - IEC 61131-3 standard defines figures and functions of FBD
  - PLCopen defines the xml formats to save programs written in languages of IEC 61131-3



Function block diagram

Background – NuDE

- Nuclear development environment (NuDE)
  - Development environment for safety-critcial software of nuclear power plant.
  - Includes tools NuSRS, NuFTA, NuSCRtoSMV, NuSCRtoFBD, FBD Tester, FBDtoVerilog, and FBD_FTA.

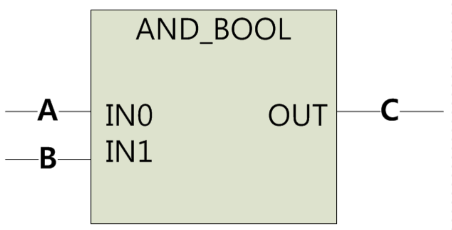


Fault tree templates

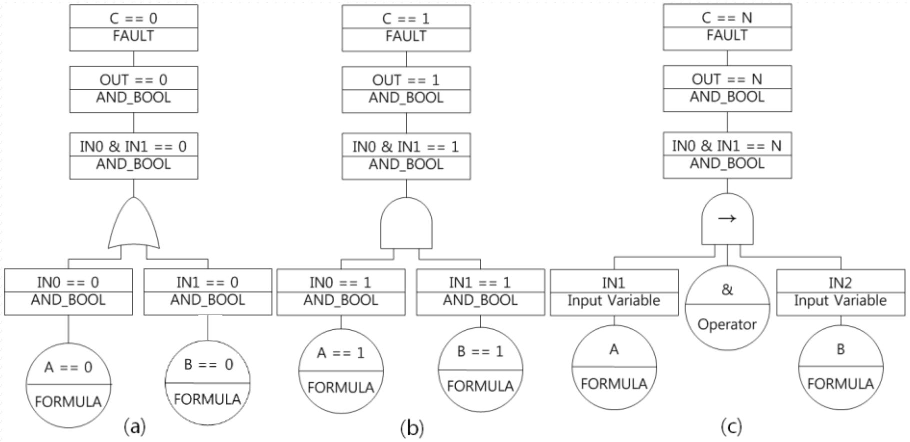
Fault Tree Templates

- Generating FT needs FT templates for each FB.
  - Each FB has their characteristics.
- Template selection is depending on FB's output.
  - The output value to three types; "0," "1," and unspecified value.
- The case "0," and "1," expressed by the connection of FB and logical gates.
  - Unspecified value is cannot represented by AND and OR gates.
- Templates can be connected to another template.
  - Connecting to leaf node of other template.
- Generate intermediate nodes for visualization.
  - Visualization for translation of fault tree.

Fault Tree Templates – Bit-string and Bitwise boolean



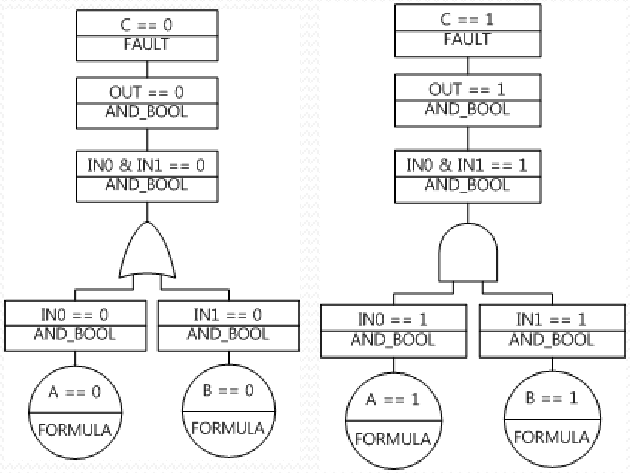
AND_BOOL function block



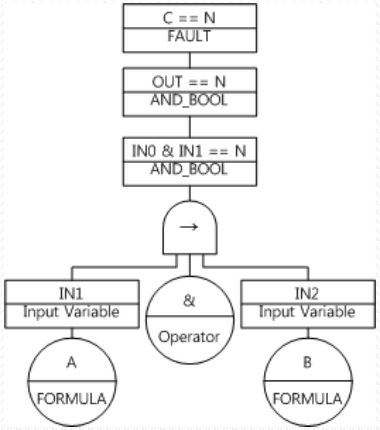
FT templates for AND_BOOL

- Bit-string and bitwise FBs
  - It used to manipulate one or more bit patterns or binary numerals for comparisons and calculations.
  - AND_BOOL behaves like a logical AND operation.
- Template for output value :
  - "0", "1" : Represented by logical gate and input variables.
  - Unspecified value : Use INORDER gate to represent the template.

Fault Tree Templates – Bit-string and Bitwise boolean



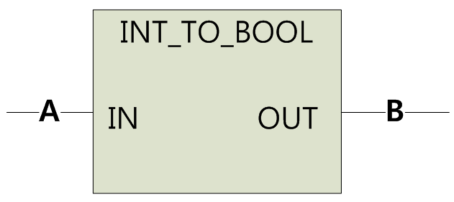
FT templates for AND_BOOL (a), (b)



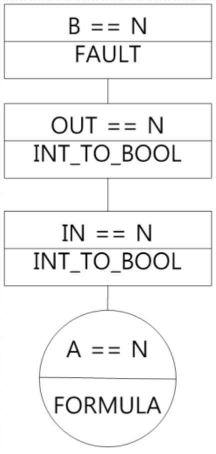
FT templates for AND_BOOL (c)

- Case A
  - Output value is 0.
  - Output is determined by one false value.
- Case B
  - Output value is 1.
  - Output needs that all input value become 1.
- Case C
  - Output value is unspecified.
  - It cannot be represented by logical gates.

Fault Tree Templates – Type conversion



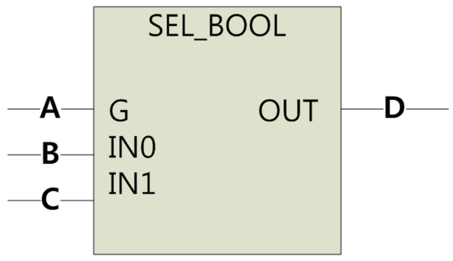
INT_TO_BOOL function block



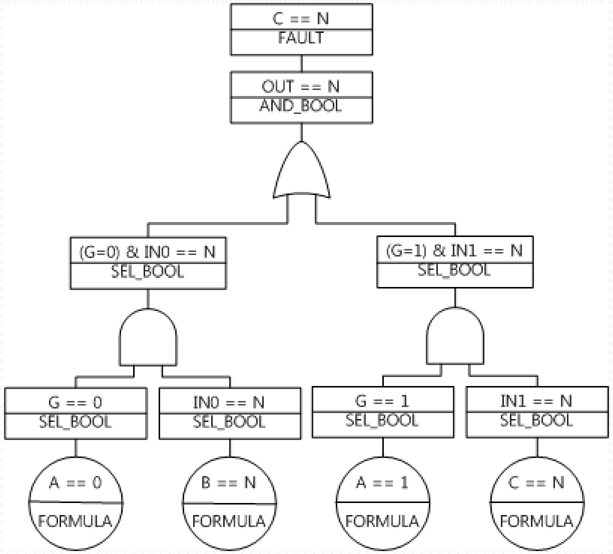
FT templates for INT_TO_BOOL

- Type conversion FBs
  - It transforms input data to another data type.
  - INT_TO_BOOL change int type value to boolean type value.
- Template for output value :
  - Always same form.
- Reflecting type conversion to formula is difficult.
  - Integer “0” and boolean “0” are same form in formula.

Fault Tree Templates – Selection and Comparison



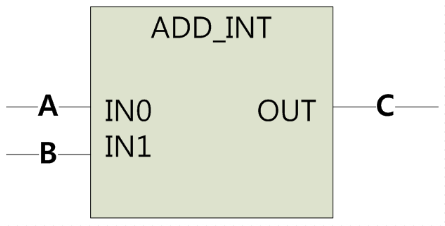
SEL_BOOL function block



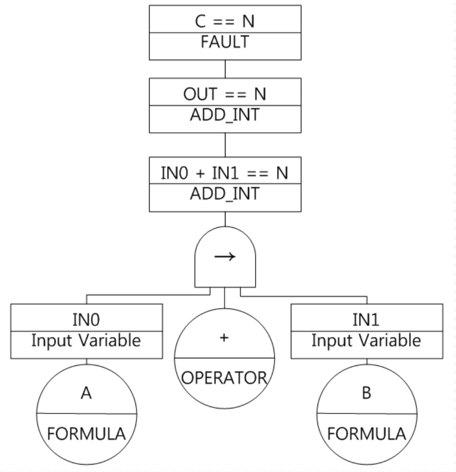
FT templates for SEL_BOOL

- Selection and comparison FBs
  - It selects one of each values between inputs.
  - SEL_BOOL outputs selectively IN0 or IN1 depending on G value.
- Template for output value :
  - Always same form.
- Regardless of the output value, selection is determined by G value.

Fault Tree Templates – Numerical



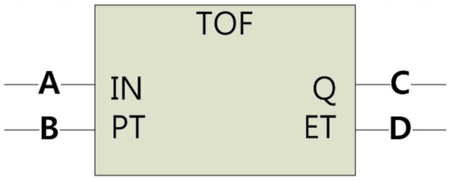
ADD_INT function block



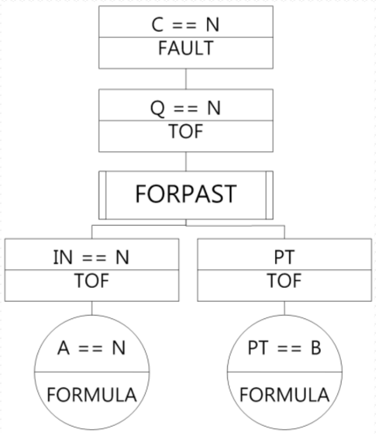
FT templates for ADD_INT

- Numerical FBs
  - It performs operations like arithmetic and trigonometric functions.
  - ADD_INT returns an output sum of the input values.
- Template for output value :
  - Always same form.
- Representing the result of ADD_INT using AND or OR gate is difficult.
  - If return value is "2," it is hard to express using "0," "1," and logical AND, OR gates.
  - Use INORDER gate to describe the template.

Fault Tree Templates – Timer



TOF function block



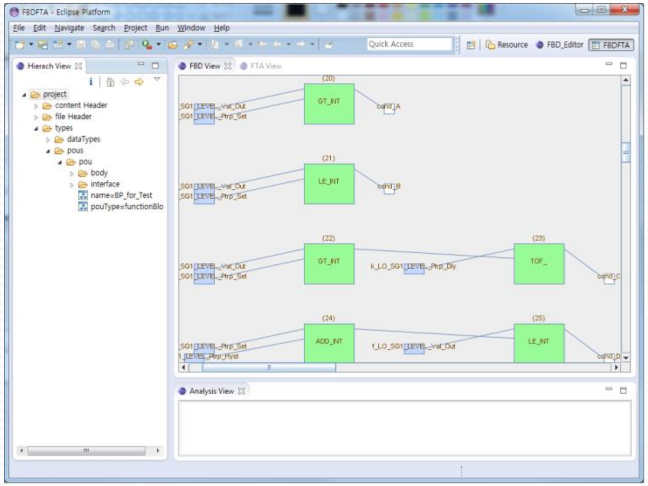
FT templates for TOF

- Timer FBs
  - It decides output value by check on the input value and determination of input value.
  - TOF receives the timer value of the whole system to ET. The value of IN is to be output to the Q value of time coming to the PT.
- Template for output value :
  - Always same form.
- For temporal properties, temporal gate is used in template.
- TFT expression

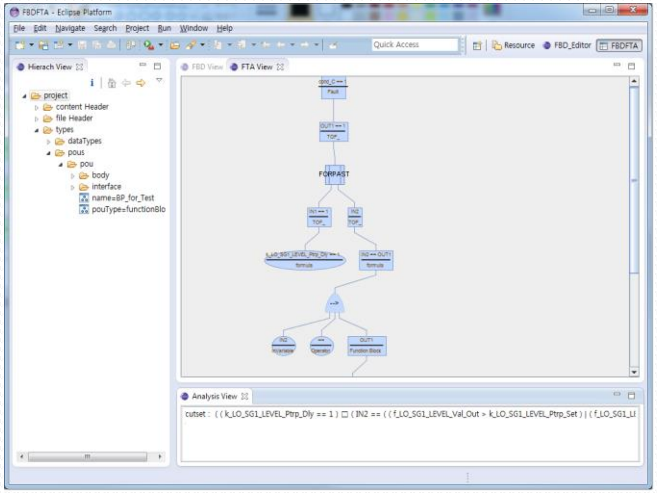
$$\boxed{(C{=}{=}1) == \Box^{-}_{B}(A{=}{=}1)}$$

FBD_FTA

FBD_FTA – Implementation of FBD_FTA



FBD View

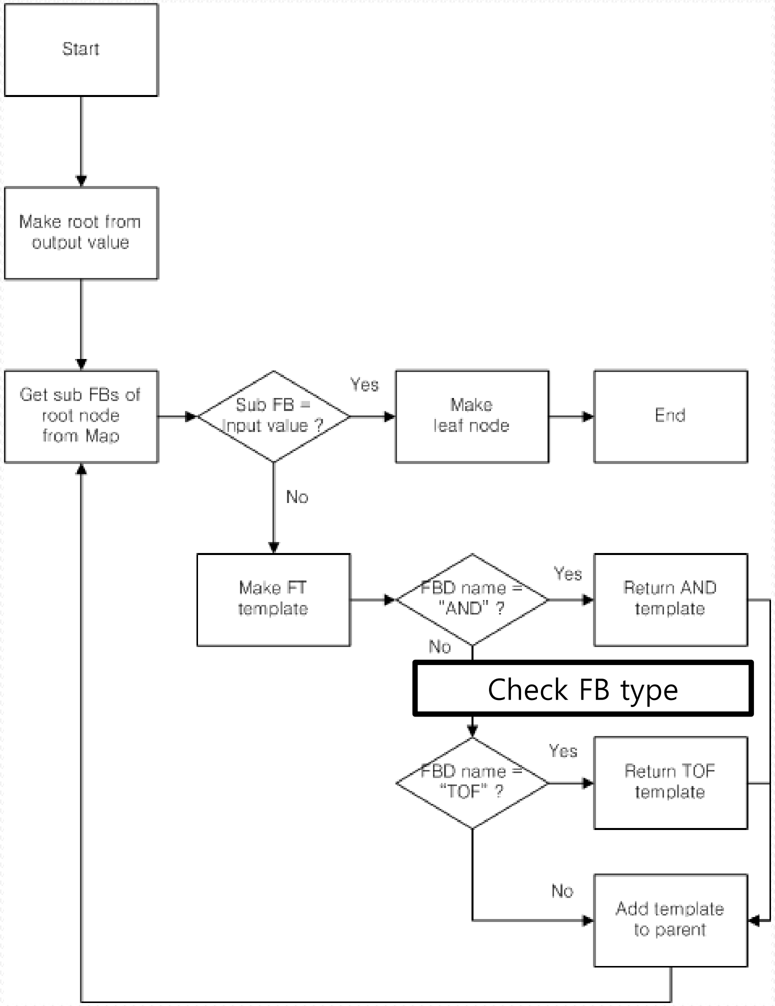


cutset : ((k_LO_SG1_LEVEL_Ptrp_Dly == 1) □ (IN2 == ((f_LO_SG1_LEVEL_Val_Out > k_LO_SG1_LEVEL_Ptrp_Set) | (f_LO_SG1_LI

FTA View

- Input :
  - Standard FBD defined in IEC 61131-3
  - Input file is generated from NuSCRtoFBD.
- Development environment :
  - Eclipse plug-in for extension and NuDE.
- Automatic generation of FT from FBD.
  - Manual generation of FT takes lots of cost and time.
- Two view for FBD and FTA.
  - FBD View shows the whole FBD program.
  - FTA View shows generated FT from selected FB in FBD View.

FBD_FTA – Generation of fault tree



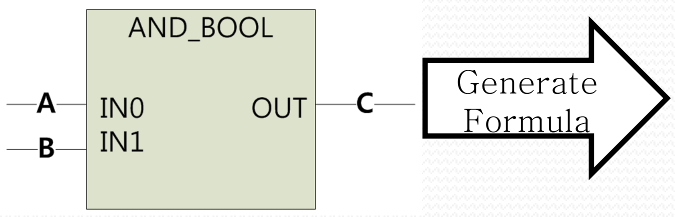
- Generate fault tree
  - Make root from output value.
  - If sub FB of current FB is not input value, FBD_FTA finds appropriate FT template and repeat this sequence recursively.
  - If sub FB of current FB is input value, FBD_FTA ends FT generation.
- Make FT Template From FBD
  - Make FT template from FB's name.
  - Use each defined template of FB.
- FBD_FTA does not care about the probabilities of nodes.
  - Generally, FTA assign probabilities to each node
  - Probability of logics in software is basically meaningless.

FBD_FTA – Finding cut-sets

- Cut-sets are another representation of FT using formula.
  - Intermediate nodes are meaningless in the formula.

- FBD_FTA find cut-sets using inorder traversal.

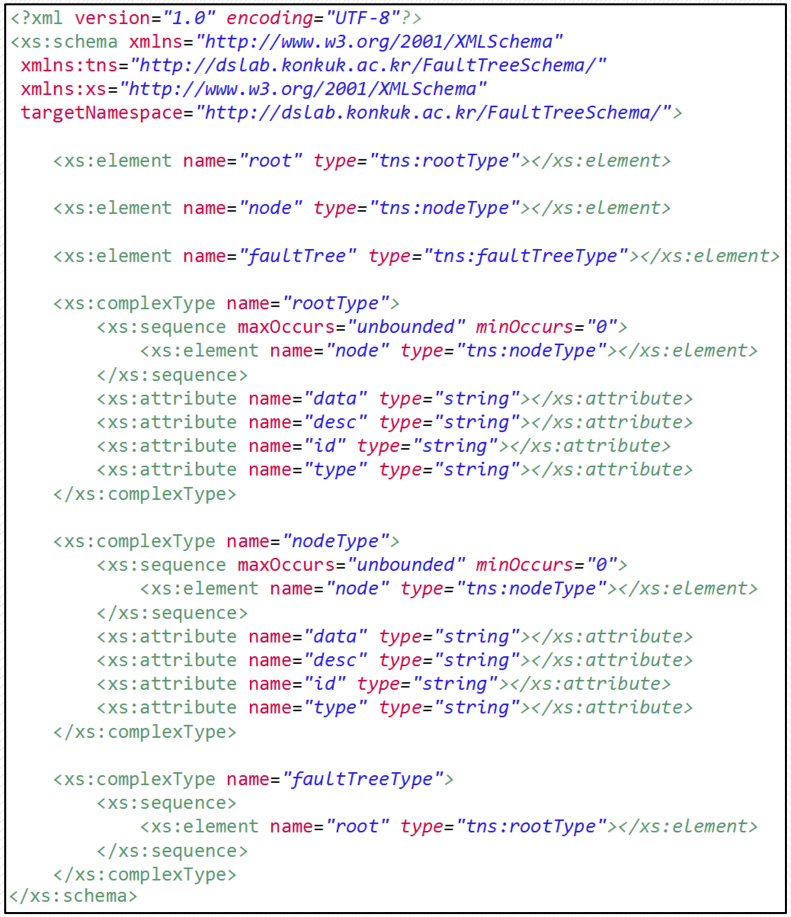
- Formula of AND_BOOL
  - If C == 1,



$$\boxed{((A == 1)\ \&\ (B == 1)) \rightarrow (C == 1)}$$

  - Input value A and B affects to the output value C.

FBD_FTA – Fault tree data structure using xml



```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns="http://www.w3.org/2001/XMLSchema"
 xmlns:tns="http://dslab.konkuk.ac.kr/FaultTreeSchema/"
 xmlns:xs="http://www.w3.org/2001/XMLSchema"
 targetNamespace="http://dslab.konkuk.ac.kr/FaultTreeSchema/">

    <xs:element name="root" type="tns:rootType"></xs:element>

    <xs:element name="node" type="tns:nodeType"></xs:element>

    <xs:element name="faultTree" type="tns:faultTreeType"></xs:element>

    <xs:complexType name="rootType">
        <xs:sequence maxOccurs="unbounded" minOccurs="0">
            <xs:element name="node" type="tns:nodeType"></xs:element>
        </xs:sequence>
        <xs:attribute name="data" type="string"></xs:attribute>
        <xs:attribute name="desc" type="string"></xs:attribute>
        <xs:attribute name="id" type="string"></xs:attribute>
        <xs:attribute name="type" type="string"></xs:attribute>
    </xs:complexType>

    <xs:complexType name="nodeType">
        <xs:sequence maxOccurs="unbounded" minOccurs="0">
            <xs:element name="node" type="tns:nodeType"></xs:element>
        </xs:sequence>
        <xs:attribute name="data" type="string"></xs:attribute>
        <xs:attribute name="desc" type="string"></xs:attribute>
        <xs:attribute name="id" type="string"></xs:attribute>
        <xs:attribute name="type" type="string"></xs:attribute>
    </xs:complexType>

    <xs:complexType name="faultTreeType">
        <xs:sequence>
            <xs:element name="root" type="tns:rootType"></xs:element>
        </xs:sequence>
    </xs:complexType>
</xs:schema>
```

- No standards of general file formats to save the result of FT.

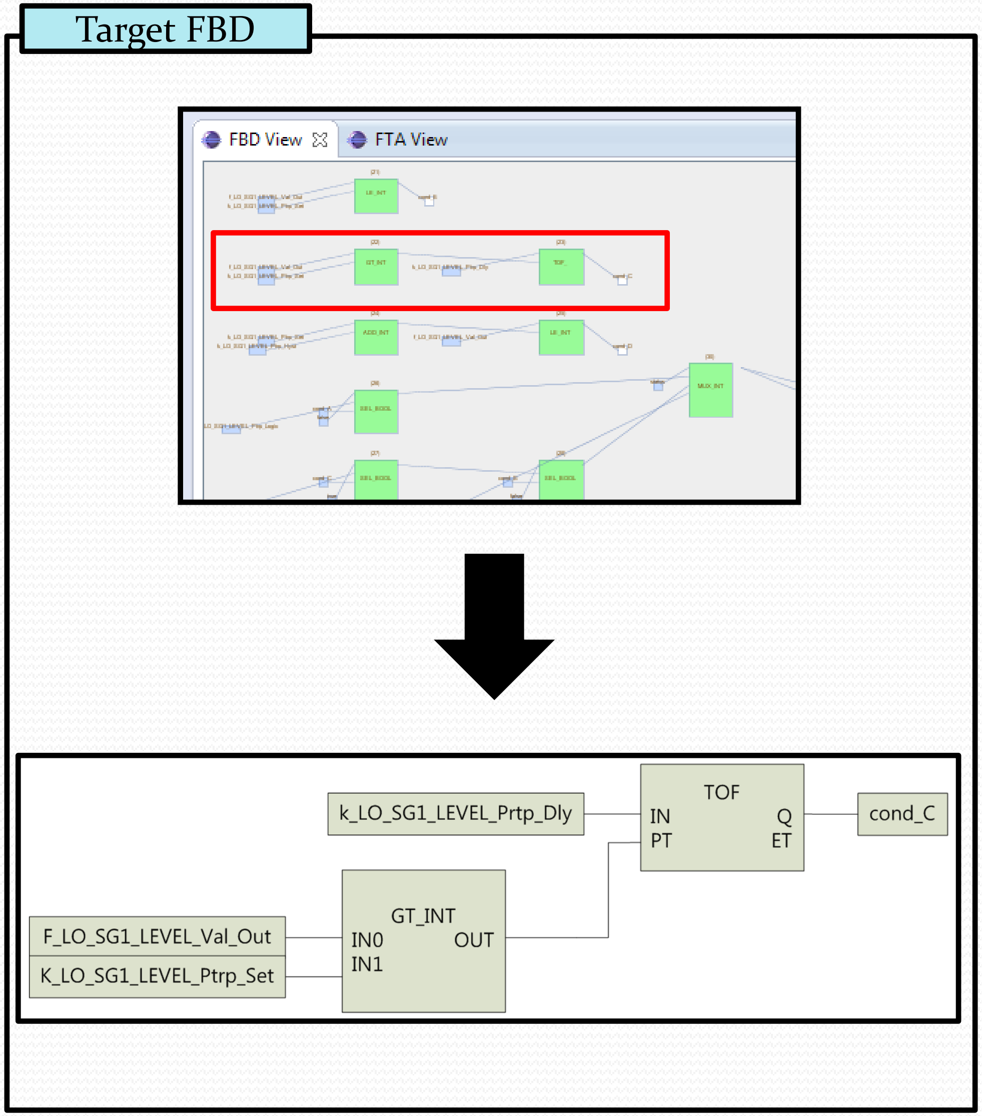
- Define xml schema for save the FT information.
  - Other tools can use this information.
  - Saving the FT helps to understand generation sequence of TF.

- Attributes
  - data : Actual information of each node in FT like formula or definition.
  - desc : Description of the information that using data only is difficult to express.
  - id : Integer value for identification and order of each node.
  - type : Gate or leaf node values for the classification of node types.

Case study

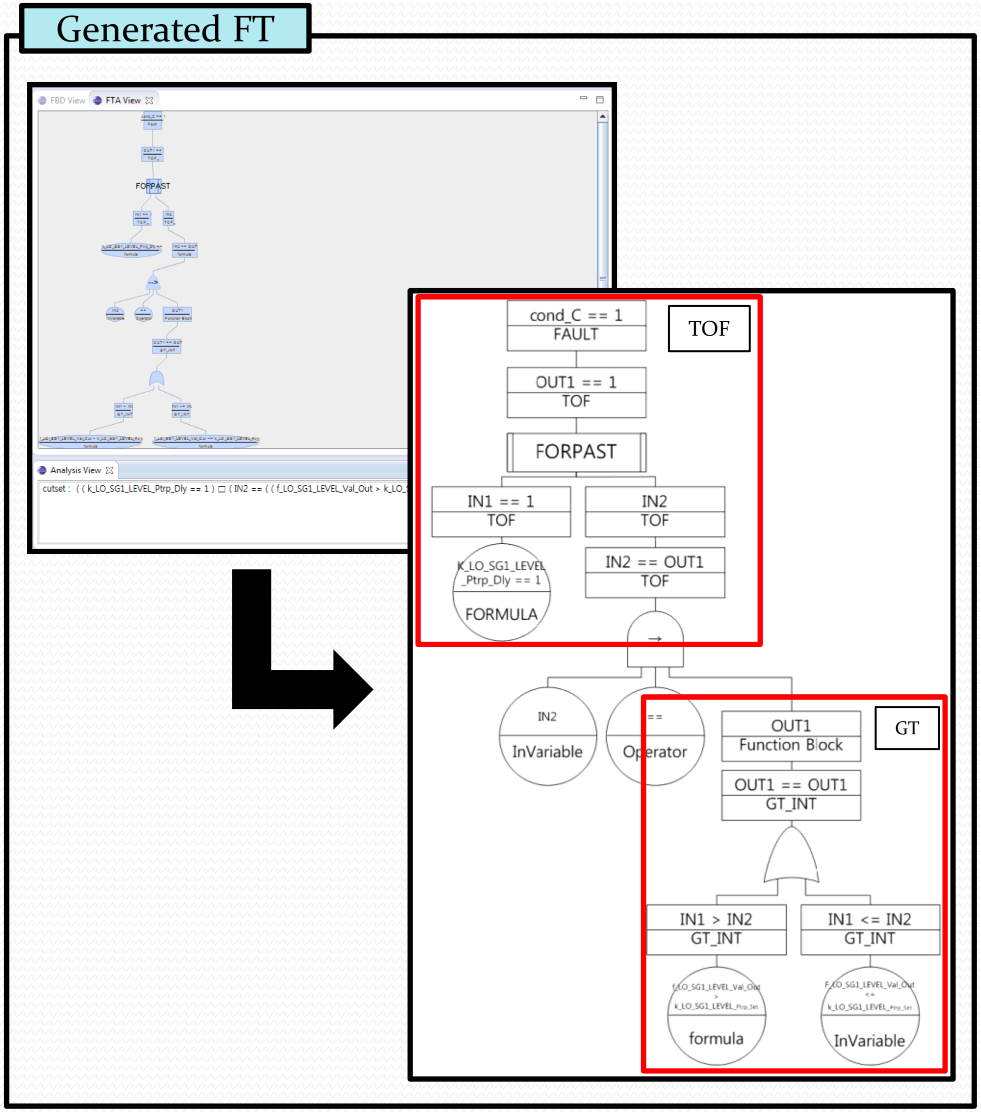
Case study

Target FBD



- Our target :
  - A part of reactor protection system (RPS) Bistable Processor (BP).
  - This program is written by FBD.
- Output
  - cond_C
- Input
  - K_LO_SG1_LEVEL_Ptrp_Dly
  - F_LO_SG1_LEVEL_Val_Out
  - K_LO_SG1_LEVEL_Ptrp_Set
- FB
  - TOF
  - GT_INT

Case study


Generated FT

cutset : ((k_LO_SG1_LEVEL_Ptrp_Dly == 1) □ (IN2 == ((f_LO_SG1_LEVEL_Val_Out > k_LO_

- Fault :
  - cond_C == 1
- Leaf nodes :
  - k_LO_SG1_LEVEL_Ptrp_Dly == 1
  - IN2 ==
  - f_LO_SG1_LEVEL_Val_Out > k_LO_SG1_LEVEL_Ptrp_Set
  - f_LO_SG1_LEVEL_Val_Out <= k_LO_SG1_LEVEL_Ptrp_Set
- Cut-sets

$$((k\_LO\_SG1\_LEVEL\_Ptrp\_Dly == 1) \square (IN2 ==$$
$$((f\_LO\_SG1\_LEVEL\_Val\_Out > k\_LO\_SG1\_LEVEL\_Ptrp\_Set) \mid$$
$$(f\_LO\_SG1\_LEVEL\_Val\_Out <= k\_LO\_SG1\_LEVEL\_Ptrp\_Set))))$$
$$\rightarrow (cond\_C == 1)$$

- Significant input value
  - k_LO_SG1_LEVEL_Ptrp_Dly
  - f_LO_SG1_LEVEL_Val_Out
  - k_LO_SG1_LEVEL_Ptrp_Set

Conclusion and Future work

Conclusion and Future work

- Conclusion
  - Generating FT from FBD needs FT templates for each FB.
  - FBD_FTA uses backward analysis to make FT and find cut-sets.
  - Automatic FT generation will help analysts perform hazard analyze of FBD program.
- Future work
  - Formalize the translated formula
  - Find minimal cut-sets

Reference

Reference

[1] Nancy Leverson. Safeware. Holt, Rinehart and Winston, 1995.

[2] D. Coppit K. Sulliva R. Manian, J. Dugan. Combining various solution techniques for dynamic fault tree analysis of computer systems. 3rd IEEE International High-Assurance Systems Engineering Symposium, 1998.

[3] G. Palshikar. Temporal fault trees. Information and Software Technology 44, 2002.

[4] S. Anderson G. Bruns. Validating safety models with fault trees. 12th International Conference on Computer Safety, Reliability, and Security, 2002.

[5] J. Dugan D. Coppit, K. Sullivan. Formal semantics of models for computational engineering: a case study on dynamic fault trees. the 11th International Symposium on Software Reliability engineering, ISSRE-2000, 2000.

[6] J. McDermid P. Fenelon. An integrated toolset for software safety analysis. Journal of Systems and Software 21 (3), 1993.

[7] R. Sasse G. Heiner Y. Papadopoulos, J. McDermid. Analysis and synthesis of the behaviour of complex programmable electronic systems in conditions of failure. Reliability Engineering and System Safety 71 (3), 2001.

[8] Junbeom Yoo Dong-Ah Lee, Eui-Sub Kim. Hazard Analysis of Function Block Diagram using Fault Tree. Korean Institute of Information Scientists and Engineers, Vol.39, 2B, 2012.

[9] IEC 61131-3 International standard. 1993.

[10] PLCopen Technical Committee 6 XML Fotmats for IEC 61131-3 Version 2.01. 2009.

[11] Jong-Hoon Lee and Junbeom Yoo. NuDE: Development Environment for Safety-Critical Software of Nuclear Power Plant. Transactions of the Korean Nuclear Society Spring Meeting 2012, 2012.

[12] NuSRS. http://dependable.kaist.ac.kr/nusrs.

[13] Junbeom Yoo Sanghyun Yun, Dong-Ah Lee. NuFTA : A CASE Tool for Automatic Software Fault Tree Analysis. Transactions of Korean Nuclear Society, vol.1, 2010.

[14] Junbeom Yoo Hyungbu Back and Sungdeok Cha. A CASE Tool for generate FBD program from Formal requirements specification of NuSCR. The Korean Institute of Information Scientists and Engineers : Computing Practices and Letters, Vol.15, 2009.

[15] Sungdeok Cha Eunkyoung Jee, Junbeom Yoo and Doohwan Bae. A Data Flow-based Structural Testing Technique for FBD Programs. Information & Software Technology, Vol.51, No.7, 2009.

[16] Sehun Jeong Junbeom Yoo, Jong-Hoon Lee and Sungdeok Cha. FBDtoVerilog: A Vendor-Independent Translation from FBDs into Verilog Programs. The Twenty-Third International Conference on Software Engineering and Knowledge Engineering (SEKE 2011), 2011.

[17] S. Cha J.S. Lee J. Yoo, T. Kim and H.S. Son. software requirements specification method for digital nuclear plants protection systems. Journal of Systems and Software, 74, 1, 2005.

[18] Zeng Shengkui Ren Yi, Liu Linlin. Fault Tree Data Structure Based on XML and the Conversion Method to BDD. World Congress on Computer Science and Information Engineering, Vol.2, 2009.

[19] KNICS (Korea Nuclear Instrumentation, Control System Research, and Development Center). <http://www.knics.re.kr/english/eindex.html>.