

Submitted to IEEE Software 2009

- Special issue on “Software Development for Embedded Systems”

Formal Modeling and Verification of Safety-Critical Software implemented in PLC

JUNBEOM YOO

KONKUK University, Korea

jbyoo@konkuk.ac.kr <http://dslab.konkuk.ac.kr>

Other Authors

Sungdeok Cha

- Professor in Korea University



Enukyoung Jee

- PhD Candidate in KAIST



Contents

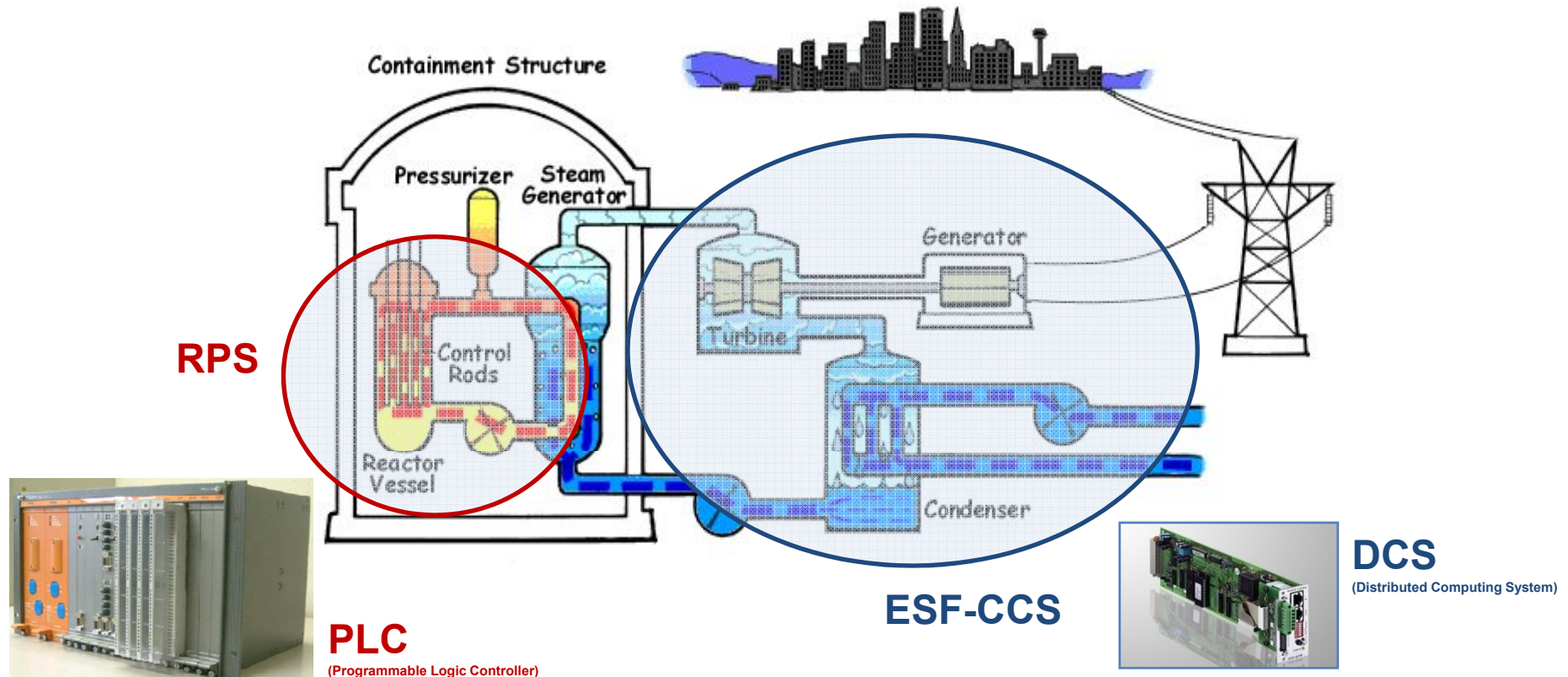
- Introduction
 - Safety-Critical Software in Nuclear Power Plants
 - Software Development Process (Existing vs. Proposed)
- Software Development Process for NPPs
 - Development Process
 - Verification Process
 - Safety Analysis Process
- Conclusion and Future Work

Introduction

1. Safety-Critical Software in Nuclear Power Plants
2. Software Development Process (Existing vs. Proposed)

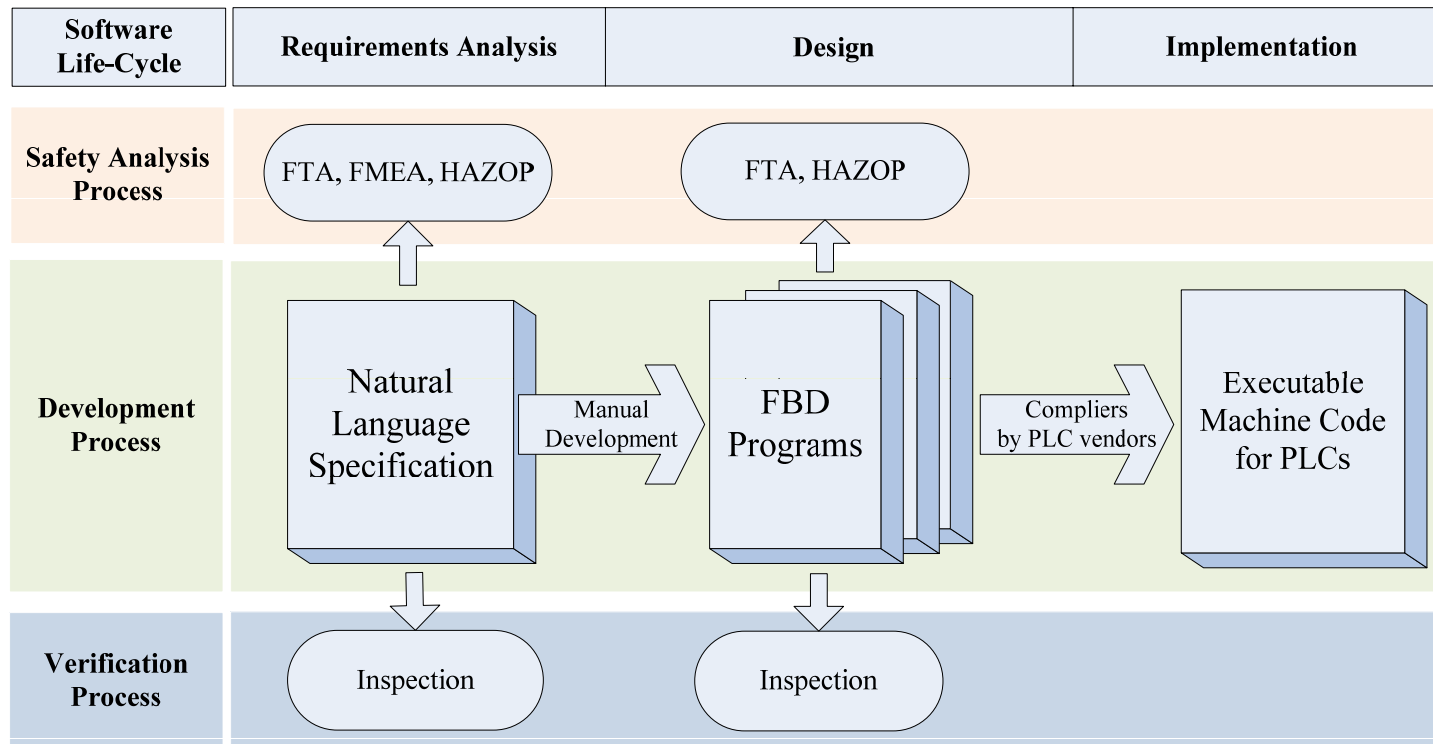
Safety-Critical Software in Nuclear Power Plants

- RPS (Reactor Protection System)
- ESF-CCS (Engineering Safety Features Component Control System)



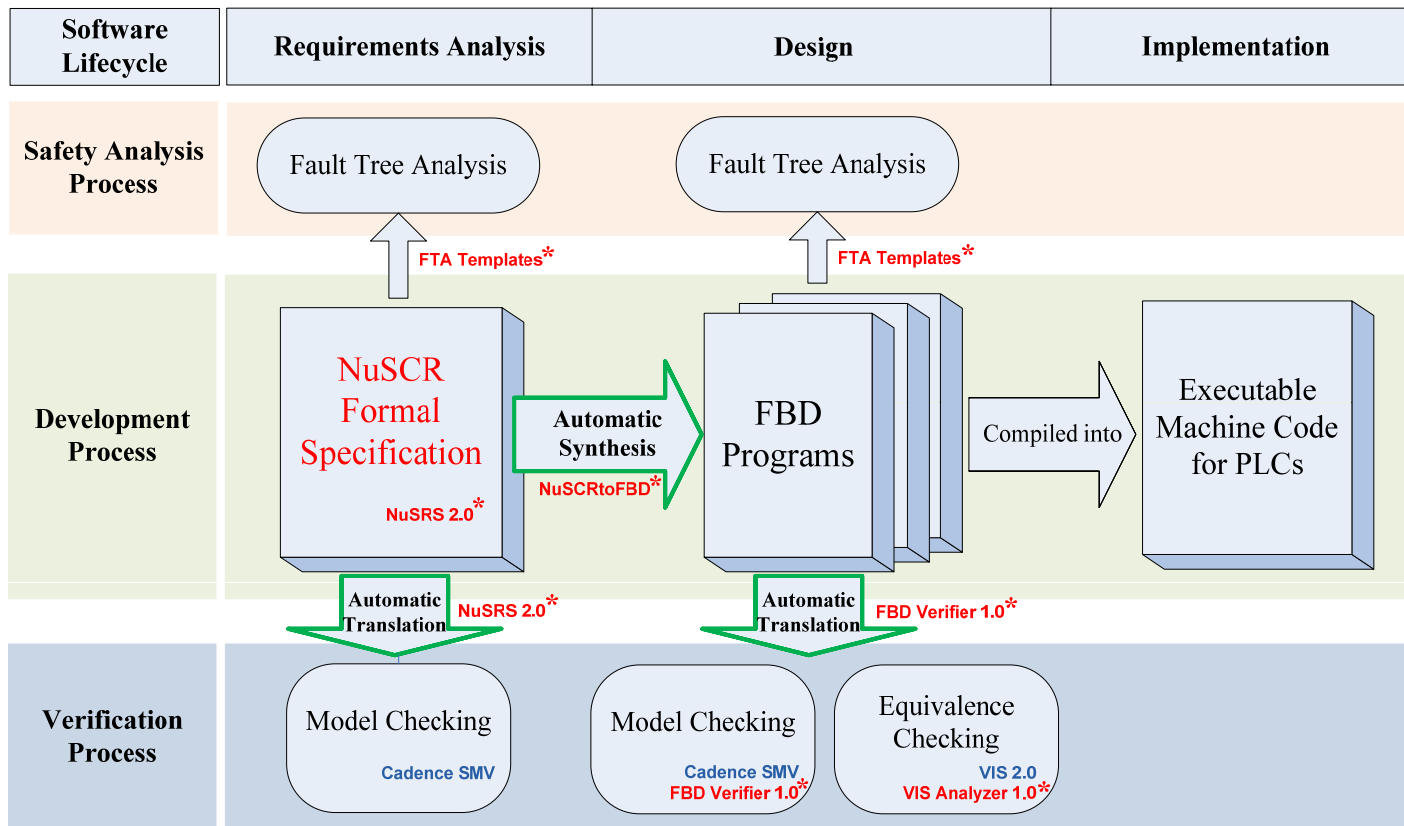
Existing Software Development Process

- For Most NPPs in Korea (e.g. Wolsung NPP)



Proposed Software Development Process

- For KNICS RPS for APR-1400 [1] (<http://www.knics.re.kr>)
 - APR-1400 : Next generation nuclear reactor being developed in Korea



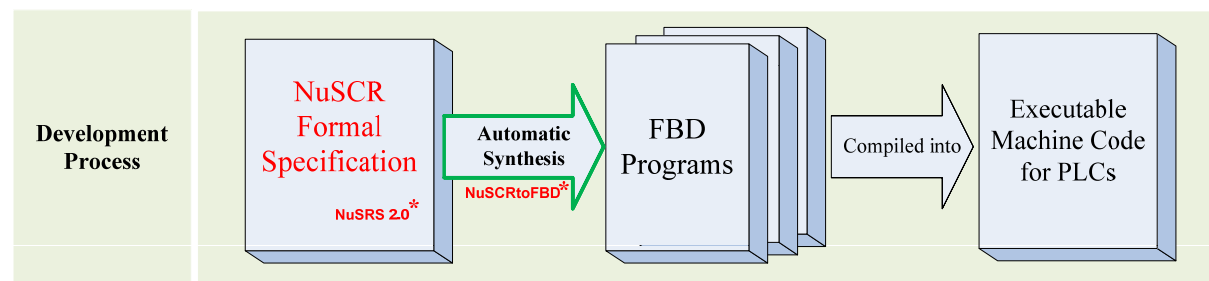
* : Our group's effort

Software Development Process for NPPs

1. Development Process
2. Verification Process
3. Safety-Analysis Process

Development Process

1. Formal Requirements Specification
2. Automatic Design Synthesis

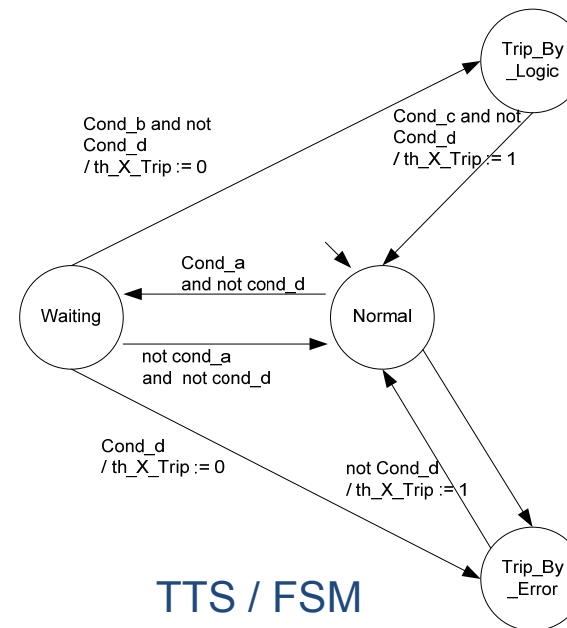


1. Formal Requirements Specification

- NuSCR [3]
 - Formal requirements specification language
 - Customized SCR [2] for nuclear applications
 - Listened to opinions offered by domain experts
- 4 constructs
 - SDT (Structured Decision Table)
 - FSM (Finite State Machine)
 - TTS (Timed Transition System)
 - FOD (Function Overview Diagram)

Conditions		
$k_X_{MIN} \leq f_X \leq k_X_{MAX}$	T	F
Actions		
$f_X_{Valid} := 0$	X	
$f_X_{Valid} := 1$		X

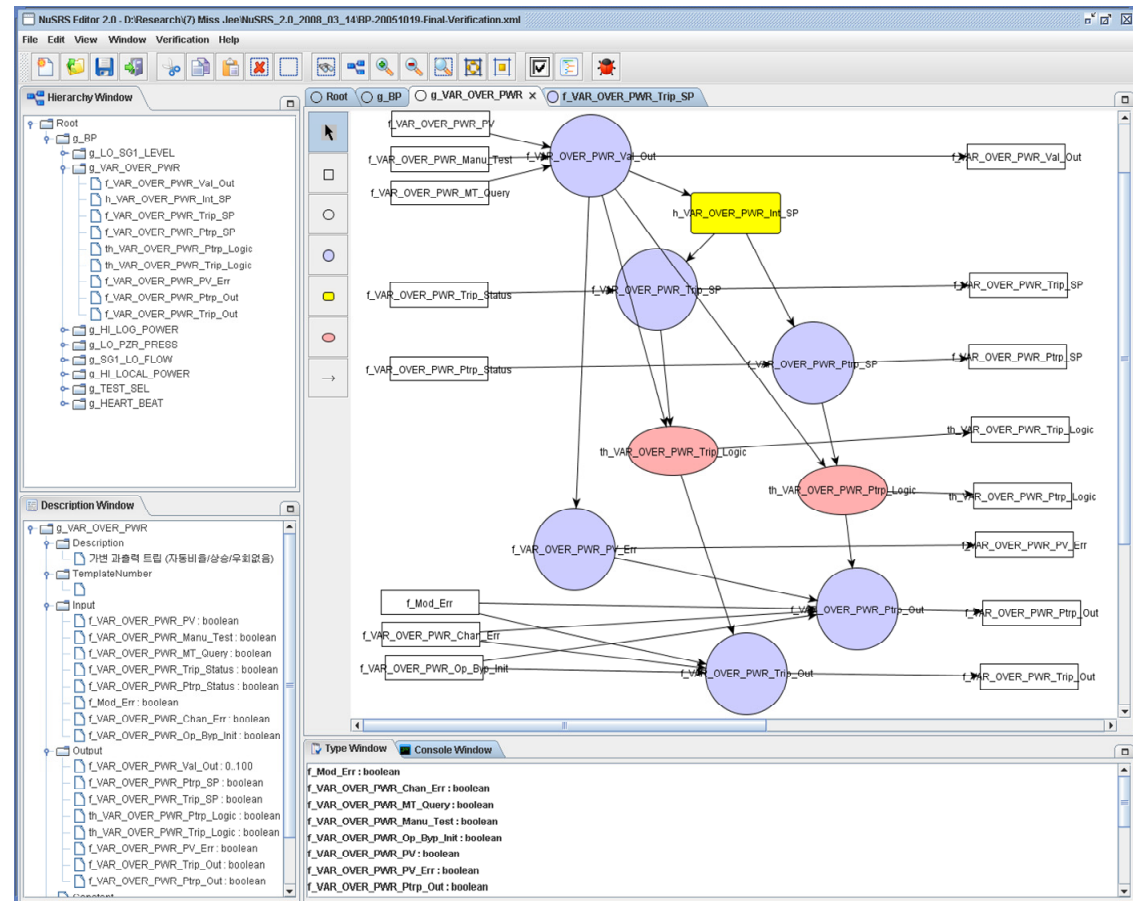
SDT



TTS / FSM

1. Formal Requirements Specification

- NuSRS (ver 2.0)
 - CASE tool supporting
 - NuSCR specification
 - Self-Checking (on-going)
 - SMV program translation (NuSCR → SMV)
 - SMV verification (CTL Model Checking)
 - Case Study
 - KNICS-RPS-SRS101, Rev,00, 2003. (by NuSRS 1.0)
 - KNICS-RPS-SVR131-01, Rev.00, 2005. (by NuSRS 2.0)

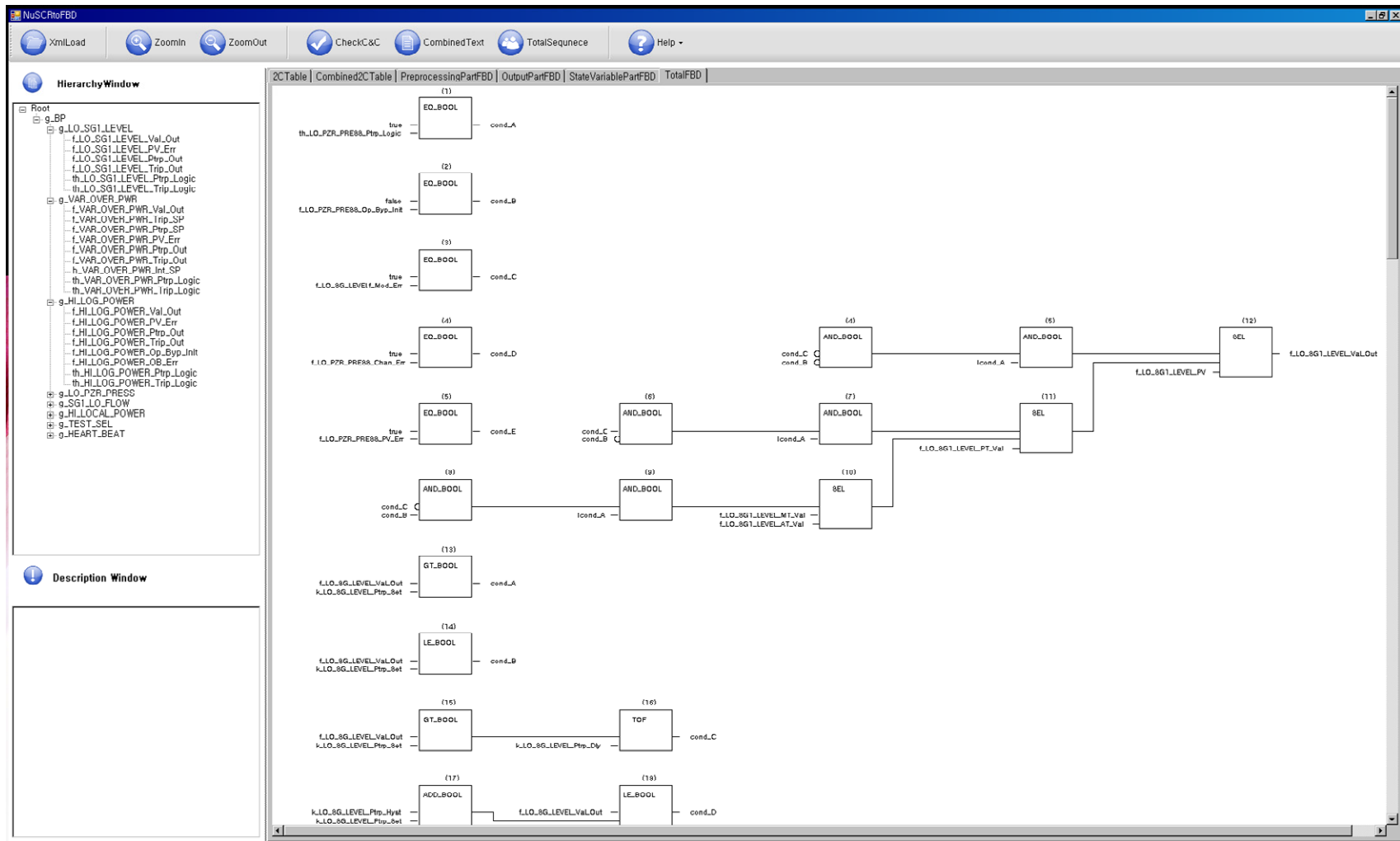


NuSRS (ver. 2.0)

2. Automatic Design Synthesis

- NuSCRtoFBD Synthesis Procedure [8]
 - Synthesizes FBD programs from NuSCR specification automatically
 - More than twice FBD blocks than manually coded and optimized ones
 - Unused in the project, because unable to develop CASE tools in advance
 - However, can be used as a baseline for FBD programming in design phase

- NuSCRtoFBD (ver 1.0)
 - CASE tool supporting
 - Automatic FBD synthesis from NuSCR
 - Reads NuSCR specification in XML format
 - Stores FBD programs in standard XML format (on-going)
 - Algorithm is being optimized

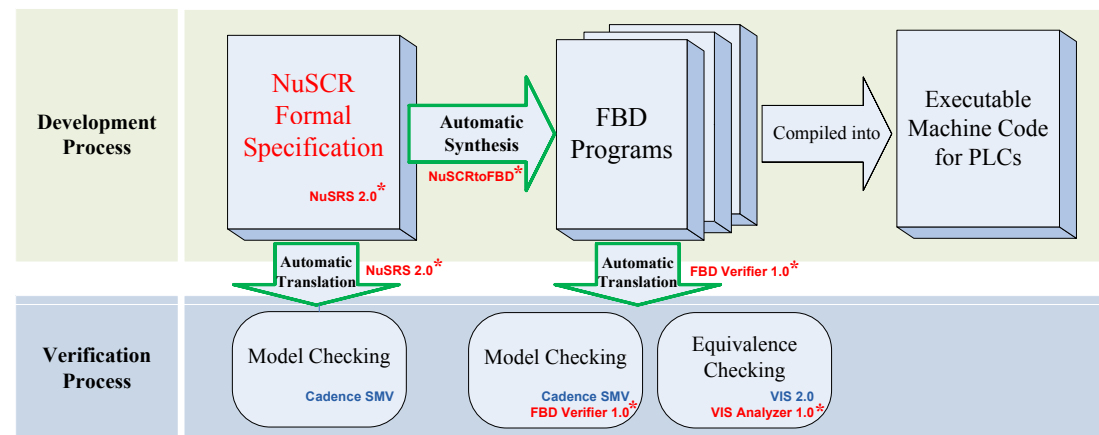


NuSCRtoFBD (ver. 1.0)

- Synthesized from KNICS RPS BP SRS (KNICS-RPS-SVR131-01, Rev.00, 2005)

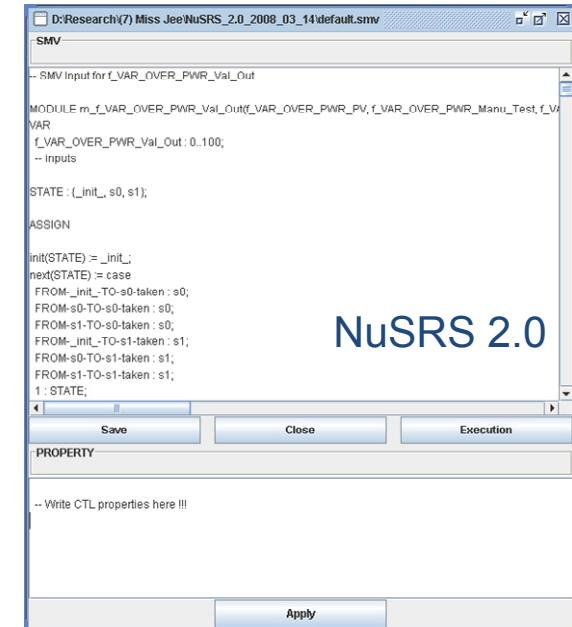
Verification Process

1. Model Checking Requirements
2. Model Checking Design
3. Equivalence Checking Designs



1. Model Checking Requirements

- Formal verification for requirements specification
 - Target : NuSCR formal specification
 - Tool : Cadence SMV [5]
 - Technique : CTL model checking
- NuSRS (ver. 2.0)
 - Automatic translation from NuSCR into SMV programs [10]
 - Seamless execution of SMV
 - Case Study
 - KNICS-RPS-SVR131-01, Rev.00, 2005
 - Found 157 errors (25 critical)



The screenshot shows a window titled "D:\Research(7) Miss Jee\NuSRS_2.0_2008_03_14\default.smv". The main text area contains SMV code for a module named "m_f_VAR_OVER_PWR_Val_Out". The code includes state declarations, an assign block, and a next function. A "PROPERTY" section is visible at the bottom, with a placeholder text "-- Write CTL properties here !!!". The interface includes buttons for "Save", "Close", "Execution", and "Apply".

```
-- SMV Input for f_VAR_OVER_PWR_Val_Out

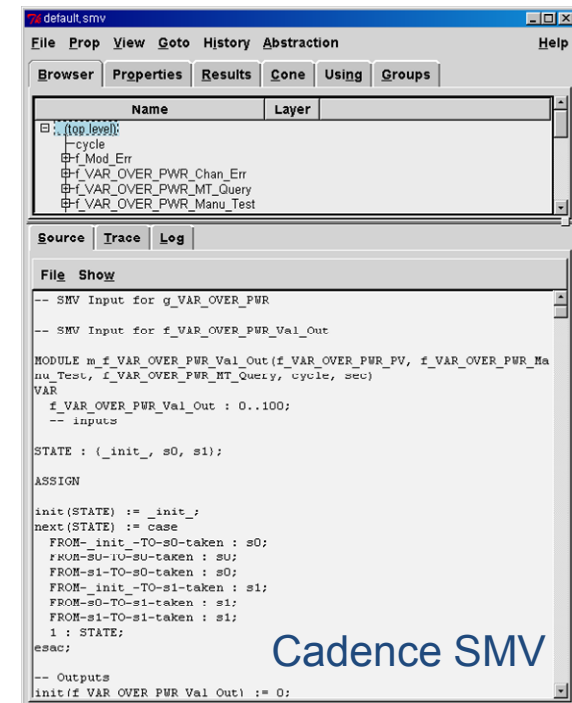
MODULE m_f_VAR_OVER_PWR_Val_Out(f_VAR_OVER_PWR_PV, f_VAR_OVER_PWR_Manu_Test, f_VAR_OVER_PWR_Val_Out)
VAR
  f_VAR_OVER_PWR_Val_Out : 0..100;
-- Inputs

STATE : (_init_, s0, s1);

ASSIGN

init(STATE) := _init_;
next(STATE) := case
  FROM _init_-TO-s0-taken : s0;
  FROM s0-TO-s0-taken : s0;
  FROM s1-TO-s0-taken : s0;
  FROM _init_-TO-s1-taken : s1;
  FROM s0-TO-s1-taken : s1;
  FROM s1-TO-s1-taken : s1;
  1 : STATE;

-- Write CTL properties here !!!
```



The screenshot shows the Cadence SMV interface. The top menu bar includes "File", "Prop", "View", "Goto", "History", "Abstraction", and "Help". Below the menu is a toolbar with "Browser", "Properties", "Results", "Cone", "Using", and "Groups". A project browser on the left shows a tree structure with "Top Level" and several sub-items. The main text area displays SMV code for a module named "m_f_VAR_OVER_PWR_Val_Out". The code includes state declarations, an assign block, and a next function. The interface includes buttons for "File", "Show", "Source", "Trace", and "Log".

```
-- SMV Input for q_VAR_OVER_PWR
-- SMV Input for f_VAR_OVER_PWR_Val_Out

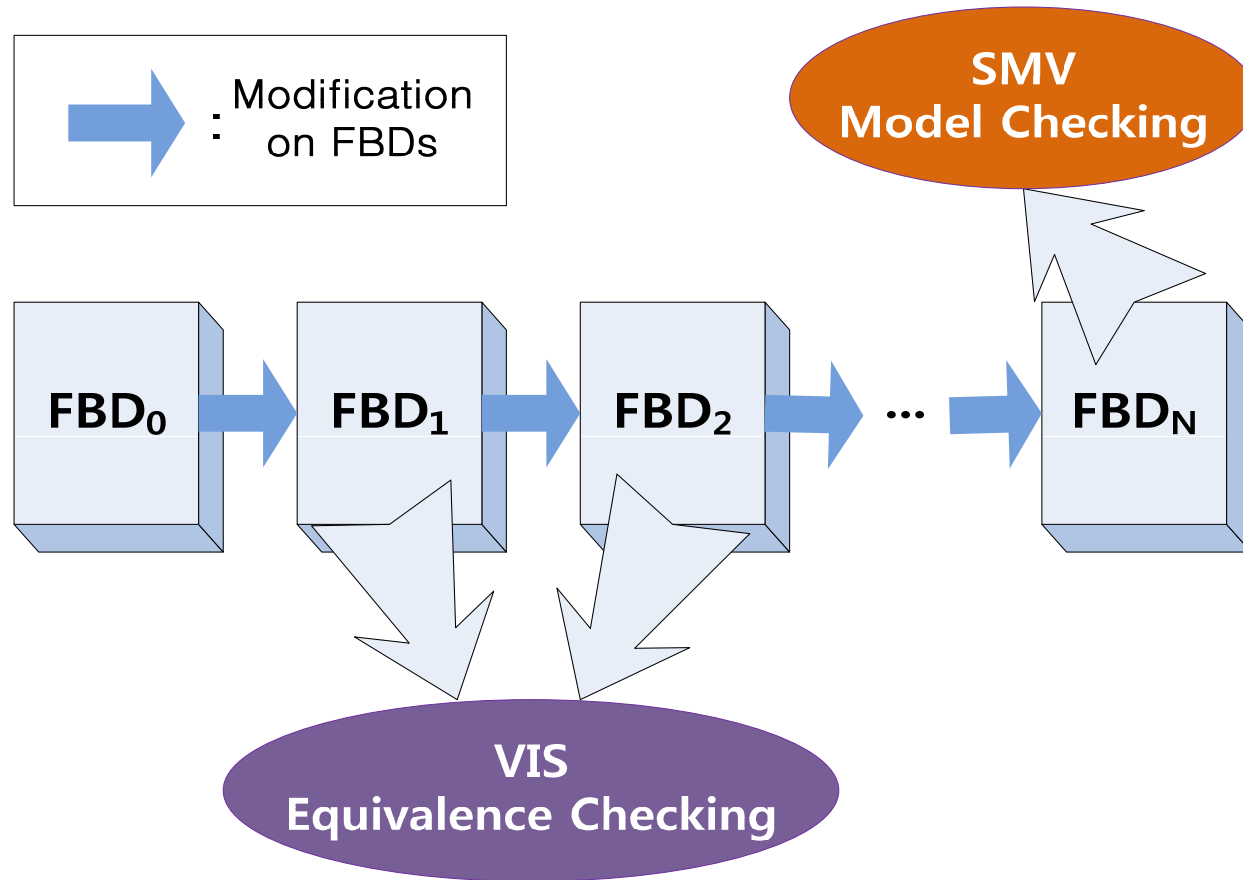
MODULE m_f_VAR_OVER_PWR_Val_Out(f_VAR_OVER_PWR_PV, f_VAR_OVER_PWR_Manu_Test, f_VAR_OVER_PWR_Val_Out, cycle, sec)
VAR
  f_VAR_OVER_PWR_Val_Out : 0..100;
-- inputs

STATE : (_init_, s0, s1);

ASSIGN

init(STATE) := _init_;
next(STATE) := case
  FROM _init_-TO-s0-taken : s0;
  FROM s0-TO-s0-taken : s0;
  FROM s1-TO-s0-taken : s0;
  FROM _init_-TO-s1-taken : s1;
  FROM s0-TO-s1-taken : s1;
  FROM s1-TO-s1-taken : s1;
  1 : STATE;
esac;

-- Outputs
init(f_VAR_OVER_PWR_Val_Out) := 0;
```



FBD Verification using
- SMV model checking & VIS Equivalence checking

2. Model Checking Design

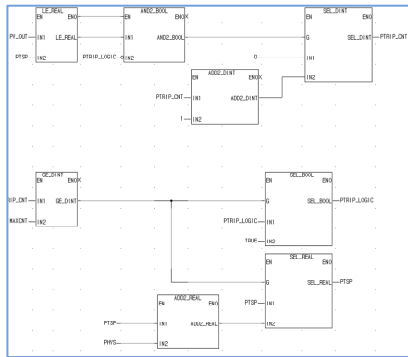
- Formal verification for design specification
 - Target : FBD program
 - Tool : Cadence SMV [5]
 - Technique : LTL model checking

- FBD Verifier (ver. 1.0 / 2.0)
 - Automatic translation from FBD programs into Verilog programs [11]
 - Seamless execution of SMV

 - Case Study
 - KNICS-RPS-SDS231, Rev.01, 2006
 - Found 60 errors (13 critical)

FBD Verifier 1.0

1. Read FBD programs in XML format



Engineering Tools by PLC vendors

2. Translate into Verilog program

```

FBD Verifier
File Verify Help
PLC
ss Jee\FBD Verifier 1.0 (200611 by jeon)\FBD2\example\FIX_RISBIN...
Verilog
D:\Research\7 Miss Jee\FBD Verifier 1.0 (200611 by jeon)\FBD2\example\FIX_RISBIN...

module main (clk, HYS, MAXCNT, PHYS, PV_OUT);
    input clk;
    input [7:0] HYS;
    input [7:0] MAXCNT;
    input [7:0] PHYS;
    reg [7:0] PTRIP_CNT;
    PTRIP_LOGIC;
    PTSP;
    [7:0] PV_OUT;
    [7:0] TRIP_CNT;
    TRIP_LOGIC;
    TSP;

    wire [8:0] PTRIP_CNT_out;
    PTRIP_LOGIC_1;
    wire [7:0] PTSP_1;
    wire [8:0] TRIP_CNT_out;
    TRIP_LOGIC_1;
    wire [7:0] TSP_1;
    TRIP_LOGIC_out;
    wire [8:0] TSP_out;
    wire [8:0] PTRIP_LOGIC_out;
    wire [8:0] PTSP_out;

    //constants
    assign HYS = 1;
    assign MAXCNT = 5;
    
```

4. Analyze verification result

3. Perform SMV model checking

```

FIX_RISBIN.v
File Prop View Goto History Abstraction Help
Browser Properties Results Cone Using Groups
Name Layer
r:\...
-PTRIP_LOGIC
-PTRIP_LOOK_1
-PTRIP_LOOK_2
-PTRIP_LOOK_3
-PTRIP_LOOK_out
@PTSP
@PTSP_1
@PTSP_2
@PTSP_3
@PTSP_out

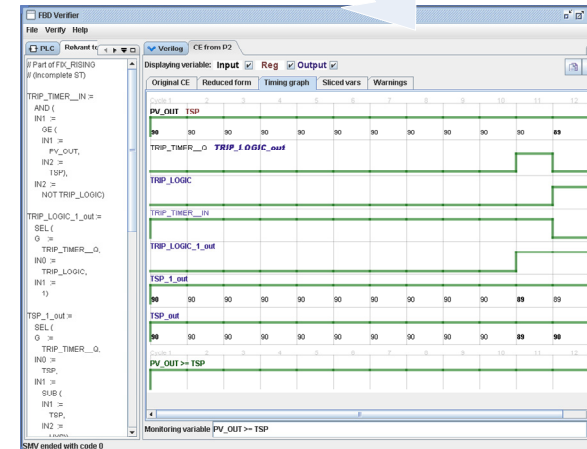
Source Trace Log
File Show
module main (clk, PV_OUT, TRIP_LOGIC_out, TSP_out, PTRIP_LOGIC_out, PTSP_out);

//constants
define HYS 0
define PHYS 0
define PTSP K 0
define TSP_F 0

input clk;
reg PTRIP_LOGIC;
reg [7:0] PTSP;
input [7:0] PV_OUT;
reg TRIP_LOGIC;
reg [7:0] TSP;

wire TRIP_LOOK_1;
wire [7:0] TRIP_LOOK_2;
wire [7:0] TRIP_LOOK_3;
wire [7:0] TRIP_LOOK_out;
output [7:0] TSP_out;
output [7:0] TSP_out;
    
```

Cadence SMV



Counterexample viewer in FBD Verifier 1.0

3. Equivalence Checking Designs

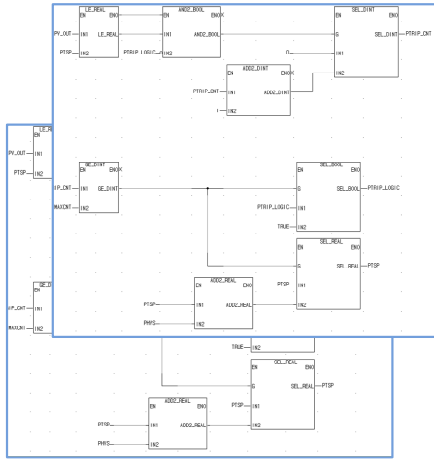
- Formal verification for design specifications
 - Target : Two FBD programs
 - Tool : VIS Verification System [4]
 - Technique : Equivalence checking, Simulation

- VIS Analyzer (ver. 1.0)
 - Seamless execution of VIS (VIS has no GUI)
 - Visualization of VIS's process and verification results [12]
 - Unused in the project, because unable to develop CASE tools in advance

 - Case Study
 - KNICS-RPS-SDS101, Rev.00, 2005
 - No official result

Trip Logic	Error Type	Compared FBD (Num. of Errors)	Original FBD (Num. of Errors)
Fixed Set-Point Rising Trip without Operating Bypass	Syntactic	0	0
	Logical	0	1
Manual Reset Variable Set-Point Trip without Operating Bypass	Syntactic	0	3
	Logical	6	2

Engineering Tools by PLC vendors



1. Read two FBD programs
in XML format

FBD Verifier 1.0

```
IN1 :=  
ADD (  
  IN1 :=  
  TSP,  
  IN2 :=  
  HV9);  
PTSP_LOGIC :=  
SEL (  
  G :=  
  AND (  
    IN1 :=  
    LE (  
      IN1 :=  
      SV_OUT,  
      IN2 :=  
      PTSP),  
    IN2 :=  
    PPRIP_LOGIC),  
  INO :=  
  PPRIP_LOGIC,  
  IN1 :=  
  0)  
PTSP :=  
SEL (  
  G :=
```

2. Translate into
Verilog programs

3. Read two Verilog programs

VIS Analyzer 1.0 (Source)

```
typedef enum (S0, S1) th_X_Pretrip_state;  
typedef enum (T0, T1, T2, T3, T4, T5, T6, T7, T8,  
T9, T10, T11, T12, T13, T14, T15, T16, T17, T18,  
T19, T20) timer_state;  
`define k_Pretrip_Setpoint 30  
`define k_X_Pretrip_Hys 10  
`define k_Trip_Delay 20  
// th_X_Pretrip module  
module th_X_Pretrip(clk, f_X, th_X_Pretrip);  
  input clk;  
  input[0:6] f_X;  
  output th_X_Pretrip;  
  //integer wire f_X;  
  //integer f_X;  
  wire th_X_Pretrip;  
  th_X_Pretrip_state reg state;  
  reg th_Prev_X_Pretrip;  
  timer_state reg timer;  
  initial state = S1;  
  initial th_Prev_X_Pretrip = 1;  
  initial timer = T0;  
  assign th_X_Pretrip = (state==S0 && f_X <=  
    k_Pretrip_Setpoint - k_X_Pretrip_Hys)?1:  
    (state==S0 && f_X >
```

VIS Analyzer 1.0 (Result)

```
C:\VC2007\Tex\home\VIS\vis-2.0\examples\RFS\FBD_Verifier\th_X_Pretrip_Manual.v  
vis> vis release 2.0 (compiled Sat Jun 14 12:02:36 2008)  
vis> vis> vis> --State 0:  
state$NFK2:S1  
state:S0  
th_Prev_X_Pretrip$NFK2:1  
th_Prev_X_Pretrip:1  
timer$NFK2:T0  
timer:T0  
--Goes to state 1:  
state:S1  
timer$NFK2:T1  
timer:T1  
--On input:  
f_X<0>:0  
f_X<1>:1  
f_X<2>:1  
f_X<3>:1  
f_X<4>:1  
f_X<5>:0  
f_X<6>:1  
--Goes to state 2:  
timer$NFK2:T2  
timer:T2  
--On input:  
<Unchanged>  
--Goes to state 3:  
timer$NFK2:T3  
timer:T3  
--On input:  
<Unchanged>  
--Goes to state 4:  
timer$NFK2:T4  
timer:T4
```

4. Perform VIS
equivalence checking

5. Verification result with
simulation

Automatic Vis Equivalence Checker

File Run Help

Verilog sources Result Result Table

# state	input	File1Output	File2Output	File1State	File2State
0	Initial	Initial	Initial	S1 1 T0	S0 1 T0
1	61	1	1	S1 1 T1	S1 1 T1
2	61	1	1	S1 1 T2	S1 1 T2
3	61	1	1	S1 1 T3	S1 1 T3
4	61	1	1	S1 1 T4	S1 1 T4
5	61	1	1	S1 1 T5	S1 1 T5
6	61	0	0	S0 0 T5	S2 0 T5
7	52	0	0	S0 0 T0	S2 0 T0
8	52	1	0	Null	Null

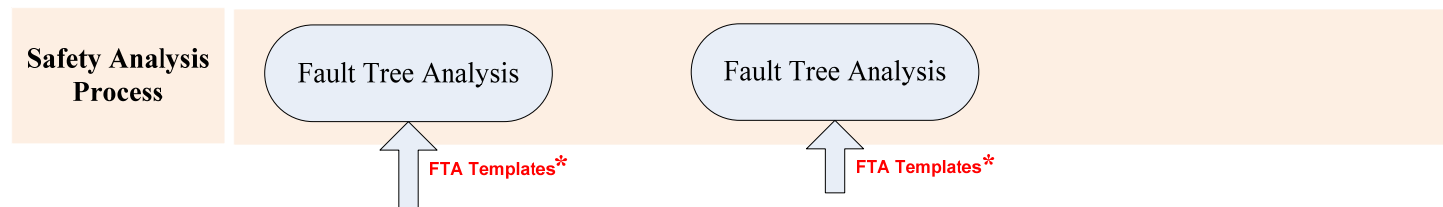
Ready

VIS Analyzer (ver. 1.0)

- Visualized and reorganized result - counterexample

Safety Analysis Process

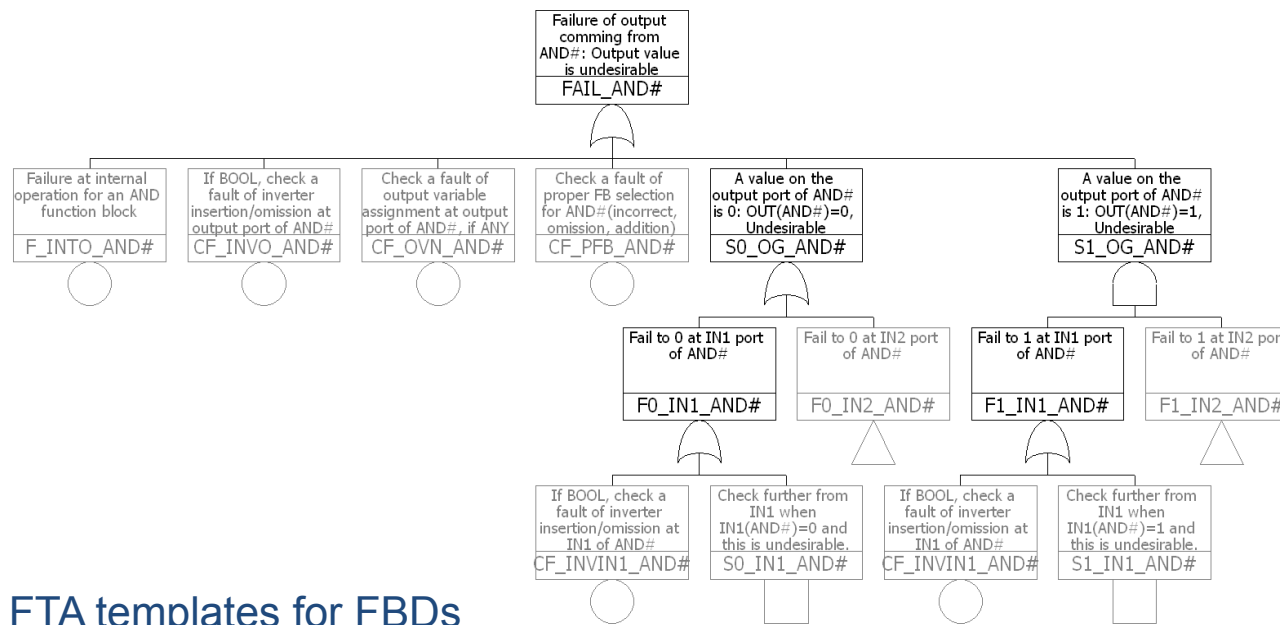
1. Fault Tree Analysis for Requirements
2. Fault Tree Analysis for Design



1. Fault Tree Analysis for Requirements

2. Fault Tree Analysis for Design

- Fault Tree Analysis
 - performed manually
 - Totally depends on analyst's experience and ability
- We provided FTA templates for NuSCR [13] and FBD [15]

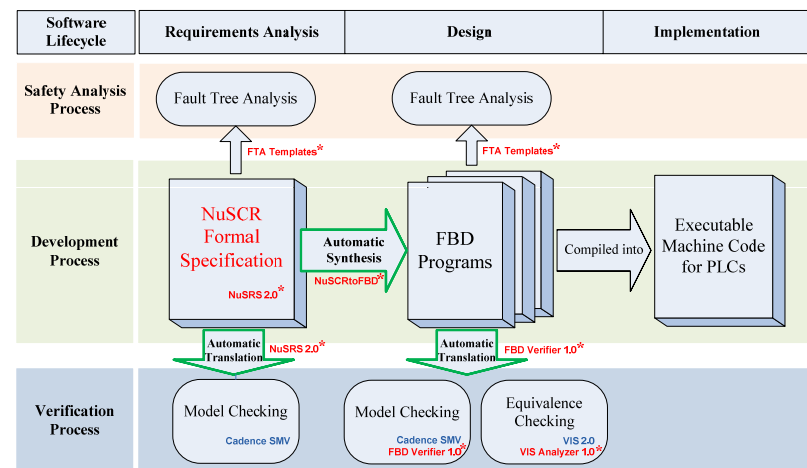


FTA templates for FBDs

Conclusion and Future Work

Conclusion

- We proposed software development processes using formal methods
 - Target: KNICS RPS for APR-1400
 - Development process
 - NuSCR formal requirements specification
 - Automatic FBD design synthesis
 - Verification process
 - Model checking NuSCR requirements
 - Model checking FBD design
 - Equivalence checking FBD designs
 - Safety analysis process
 - FTA templates for NuSCR requirements
 - FTA templates for FBD programs
 - Case Study
 - KNICS-RPS-SVR131-01, Rev.00, 2005
 - KNICS-RPS-SDS231, Rev.01, 2006



*: Our Group's effort

Future Work

1. Integrated Tool-set

2. Tool Enhancement

- Self-checking : completeness & consistency (NuSRS)
- Synchronous Verilog issue in model checking FBD programs using SMV (FBD Verifier)
- Optimization of FBD synthesis algorithm (NuSCRtoFBD)
- Add other functions to VIS Analyzer (VIS Analyzer)

3. Traceability Analysis

- From requirements to design
- From requirements' FTA to design's FTA

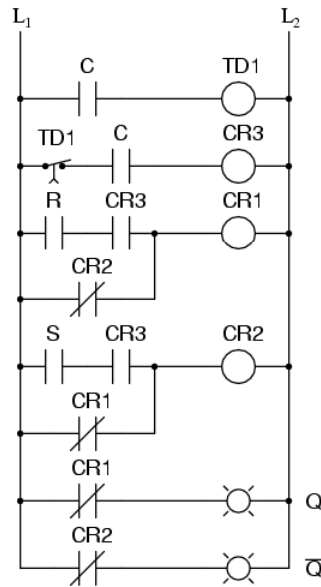
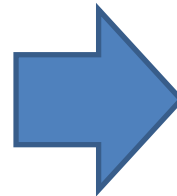
4. FBD Testing

- Measures (coverage criteria)
- Testing tool support

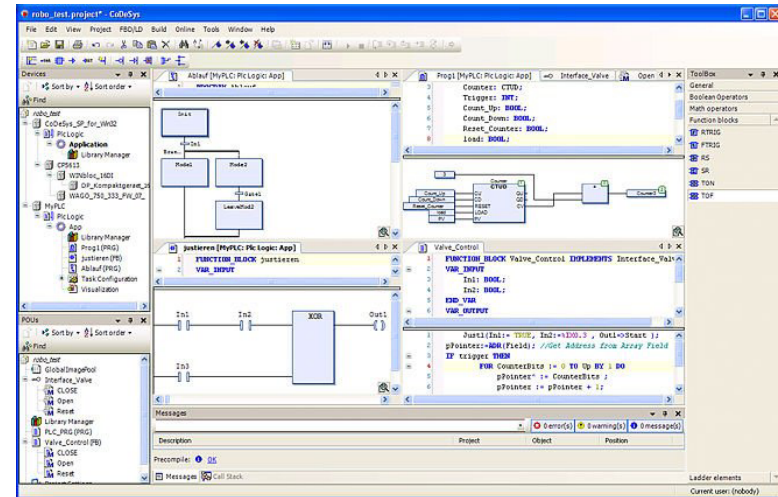
5. Application to Other Domains

References

- [1] KNICS (Korea Nuclear Instrumentation & Control System R&D Center). <http://www.knics.re.kr>.
- [2] Kathryn L. Heninger, "Specifying Software Requirements for Complex Systems: New Techniques and Their Application," *IEEE Transactions on Software Engineering*, SE Vol.6, No.1, pp2-13, 1980.
- [3] Junbeom Yoo, Taihyo Kim, Sungdeok Cha, Jang-Su Lee, Han Seong Son, "A Formal Software Requirements Specification Method for Digital Nuclear Plants Protection Systems," *Journal of Systems and Software*, Vol.74, No.1, pp.73-83, 2005.
- [4] VIS (Verification Interacting with Synthesis), [http:// embedded.eecs.berkeley.edu/research/vis](http://embedded.eecs.berkeley.edu/research/vis).
- [5] SMV (Symbolic Model Verifier), <http://www.kenmcmil.com/smv.html>.
- [6] Sungdeok Cha, "Pet Formalisms versus Industry-Proven Survivors: Issues on Formal Methods Education," *Journal of Research and Practice in Information Technology*, Vol.32, No.1, pp39-46, 2000.
- [7] Mats P.E. Heimdahl and Nancy G. Leveson, "Completeness and Consistency in Hierarchical State-Based Requirements," *IEEE Transactions on Software Engineering*, Vol.22, No.6, pp363-377, 1996.
- [8] Junbeom Yoo, Sungdeok Cha, Chang Hwoi Kim, Duck Yong Song, "Synthesis of FBD-based PLC Design from NuSCR Formal Specification," *Reliability Engineering and System Safety*, Vol.87, No.2, pp287-294, 2005.
- [9] US NRC, Digital Instrumentation and Control Systems in Nuclear Power Plants: Safety and Reliability Issues, *National Academy Press*, 1997
- [10] Jaemyung Cho, Junbeom Yoo, Sungdeok Cha, "NuEditor – A Tool Suite for Specification and Verification of NuSCR," In proceeding of *Second ACIS International Conference on Software Engineering Research, Management and Applications (SERA2004)*, pp298-304, LA, USA, May 5-7, 2004.
- [11] Junbeom Yoo, Sungdeok Cha, and Eunkyong Jee, "A Verification Framework for FBD based Software in Nuclear Power Plants," In the proceeding of *15th Asia Pacific Software Engineering Conference (APSEC)*, pp.385-392, Beijing, China, Dec. 3-5, 2008.
- [12] Junbeom Yoo, Sungdeok Cha, and Eunkyong Jee, "Verification of PLC Programs written in FBD with VIS," *Nuclear Engineering and Technology*, Vol.41, No.2, 2009, to be published.
- [13] Taeho Kim, Junbeom Yoo, Sungdeok Cha, "A Synthesis Method of Software Fault Tree from NuSCR Formal Specification using Templates," *Journal of Korea Institute of Information Scientists and Engineers (in Korean)*, SE Vol.32, No.12, pp1178-1192, 2005.
- [14] Younju Oh, Junbeom Yoo, Sungdeok Cha, Han Seong Son, "Software Safety Analysis of Function Block Diagrams using Fault Trees," *Reliability Engineering and System Safety*, Vol.88, No.3, pp215-228, 2005.
- [15] Gee-Yong Park, Kwang Yong Koh, Eunkyong Jee, Poong Hyun Seong, Kee-Choon Kwon and Dae Hyung Lee, "Fault Tree Analysis of KNICS RPS Software," *Nuclear Engineering and Technology*, Vol.40, No.5, pp397-408, 2008.



C	S	R	Q	Q̄
┌	┌	┌	latch	latch
┌	┌	┌	0	1
┌	┌	┌	1	0
┌	┌	┌	0	0
x	┌	┌	latch	latch
x	┌	┌	latch	latch
x	┌	┌	latch	latch
x	┌	┌	latch	latch



Relay-based Analog System

PLC-based Digital System

NuSRS Editor 2.0 - D:\Research(7) Miss_Jee\NuSRS_2.0_2008_03_14\RP.20051019.Final.Verification.xml

File Edit View Window Verification Help

Hierarchy Window

- Root
 - g_BP
 - g_LO_SG1_LEVEL
 - g_VAR_OVER_PWR
 - f_VAR_OVER_PWR_Val_Out
 - h_VAR_OVER_PWR_Int_SP
 - f_VAR_OVER_PWR_Val_Out
 - h_VAR_OVER_PWR_Int_SP
 - f_VAR_OVER_PWR_Val_Out
 - f_VAR_OVER_PWR_Val_Out
 - f_VAR_OVER_PWR_Val_Out
 - f_VAR_OVER_PWR_Manu_Test
 - f_VAR_OVER_PWR_Val_Out
 - f_VAR_OVER_PWR_Mt_Query
 - f_VAR_OVER_PWR_Val_Out
 - f_VAR_OVER_PWR_Trip_Status
 - f_VAR_OVER_PWR_Trip_SP
 - f_VAR_OVER_PWR_Ptrip_Status
 - f_VAR_OVER_PWR_Ptrip_SP
 - th_VAR_OVER_PWR_Trip_Logic
 - th_VAR_OVER_PWR_Trip_Logic
 - th_VAR_OVER_PWR_Ptrip_Logic
 - th_VAR_OVER_PWR_Ptrip_Logic
 - f_VAR_OVER_PWR_PV_Err
 - f_VAR_OVER_PWR_PV_Err
 - f_VAR_OVER_PWR_Ptrip_Out
 - f_VAR_OVER_PWR_Ptrip_Out
 - f_VAR_OVER_PWR_Trip_Out
 - f_VAR_OVER_PWR_Trip_Out

Description Window

g_VAR_OVER_PWR

- Description
 - 가변 과충력 트립 (자동비출/상승/우회없음)
- TemplateNumber
- Input
 - f_VAR_OVER_PWR_PV : boolean
 - f_VAR_OVER_PWR_Manu_Test : boolean
 - f_VAR_OVER_PWR_Mt_Query : boolean
 - f_VAR_OVER_PWR_Trip_Status : boolean
 - f_VAR_OVER_PWR_Ptrip_Status : boolean
 - f_Mod_Err : boolean
 - f_VAR_OVER_PWR_Chan_Err : boolean
 - f_VAR_OVER_PWR_Op_By_Invit : boolean
- Output
 - f_VAR_OVER_PWR_Val_Out : 0..100
 - f_VAR_OVER_PWR_Trip_SP : boolean
 - f_VAR_OVER_PWR_Ptrip_SP : boolean
 - th_VAR_OVER_PWR_Trip_Logic : boolean
 - th_VAR_OVER_PWR_Ptrip_Logic : boolean
 - f_VAR_OVER_PWR_PV_Err : boolean
 - f_VAR_OVER_PWR_Trip_Out : boolean
 - f_VAR_OVER_PWR_Ptrip_Out : boolean

Type Window

- f_Mod_Err : boolean
- f_VAR_OVER_PWR_Chan_Err : boolean
- f_VAR_OVER_PWR_Mt_Query : boolean
- f_VAR_OVER_PWR_Manu_Test : boolean
- f_VAR_OVER_PWR_Op_By_Invit : boolean
- f_VAR_OVER_PWR_PV : boolean
- f_VAR_OVER_PWR_PV_Err : boolean
- f_VAR_OVER_PWR_Trip_Out : boolean
- f_VAR_OVER_PWR_Ptrip_Out : boolean

default.smv

File Prop View Goto History Abstraction Help

Browser Properties Results Cone Using Groups

Name	Layer
cycle	
f_Mod_Err	
f_VAR_OVER_PWR_Chan_Err	
f_VAR_OVER_PWR_Mt_Query	
f_VAR_OVER_PWR_Manu_Test	
f_VAR_OVER_PWR_Op_By_Invit	
f_VAR_OVER_PWR_PV	
f_VAR_OVER_PWR_PV_Err	
f_VAR_OVER_PWR_Ptrip_Out	

Source Trace Log

File Show

```

-- SMV Input for g_VAR_OVER_PWR
-- SMV Input for f_VAR_OVER_PWR_Val_Out
MODULE m f_VAR_OVER_PWR_Val_Out(f_VAR_OVER_PWR_PV, f_VAR_OVER_PWR_Manu_Test, f_VAR_OVER_PWR_Mt_Query, cycle, sec)
VAR
  f_VAR_OVER_PWR_Val_Out : 0..100;
  -- inputs
STATE : (_init_, s0, s1);
ASSIGN
  init(STATE) := _init_;
  next(STATE) := case
    FROM _init_ TO s0 :

```

Dr:\Research(7) Miss_Jee\NuSRS_2.0_2008_03_14\default.smv

SMV

```

DEFINE
-- Constants
false = 0;
true = 1;
--(in-state_name = state = state_name)
in_init = STATE = _init_;
in_s0 = STATE = s0;
--(transition_name-enabled = in-state_name & enable_cond)
FROM _init_ TO s0-enabled = in_init & ((f_VAR_OVER_PWR_Trip_Status = false));
FROM s0 TO s0-enabled = in_s0 & ((f_VAR_OVER_PWR_Trip_Status = false));
--(transition_name-taken = transition_name-enabled)
FROM _init_ TO s0-taken = FROM _init_ TO s0-enabled;
FROM s0 TO s0-taken = FROM s0 TO s0-enabled;

```

Save Close Execution

PROPERTY

Apply

NuSRS 2.0

- Full specification for KNICS RPS BP SRS (KNICS-RPS-SVR131-01, Rev.00, 2005)

Target subsystems		BP	CP
System Information	#pages of natural lang. spec.	60	46
	#group nodes of NuSCR spec.	8	10
	#nodes of NuSCR spec.	81	111
Verification Information	#properties	124	207
Detected Errors	Incorrect specification	1	0
	Omission	12	6
	Logic check required	6	0
	Incorrect formal specification	2	2
	Syntax error of formal specification	83	45
Total #errors		104	53
Primary #errors		19	6

SMV Verification Result

- KNICS RPS BP & CP SRS (KNICS-RPS-SVR131-01, Rev.00, 2005)

Target subsystems		BP	CP
System Information	#pages of natural lang. spec.	190	163
	#function blocks	1,335	1,623
	#variables	1,038	820
	#lines of Verilog model	7,862	3,085
Verification Information	#properties	216	83
Detected Errors	Incorrect logic	14	6
	Omission	0	2
	Incorrect in certain condition	4	0
	Incorrect FBD	13	5
	Incorrect SDS	16	0
Total #errors		47	13
Distinct #errors		10	3

SMV Verification Result

- KNICS RPS BP & CP SDS (KNICS-RPS-SDS231, Rev.01, 2006)