

정형 요구사항 명세 기반 원자력 소프트웨어 개발 방법론

Dependable Software Laboratory

KONKUK University, Korea

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

Contents

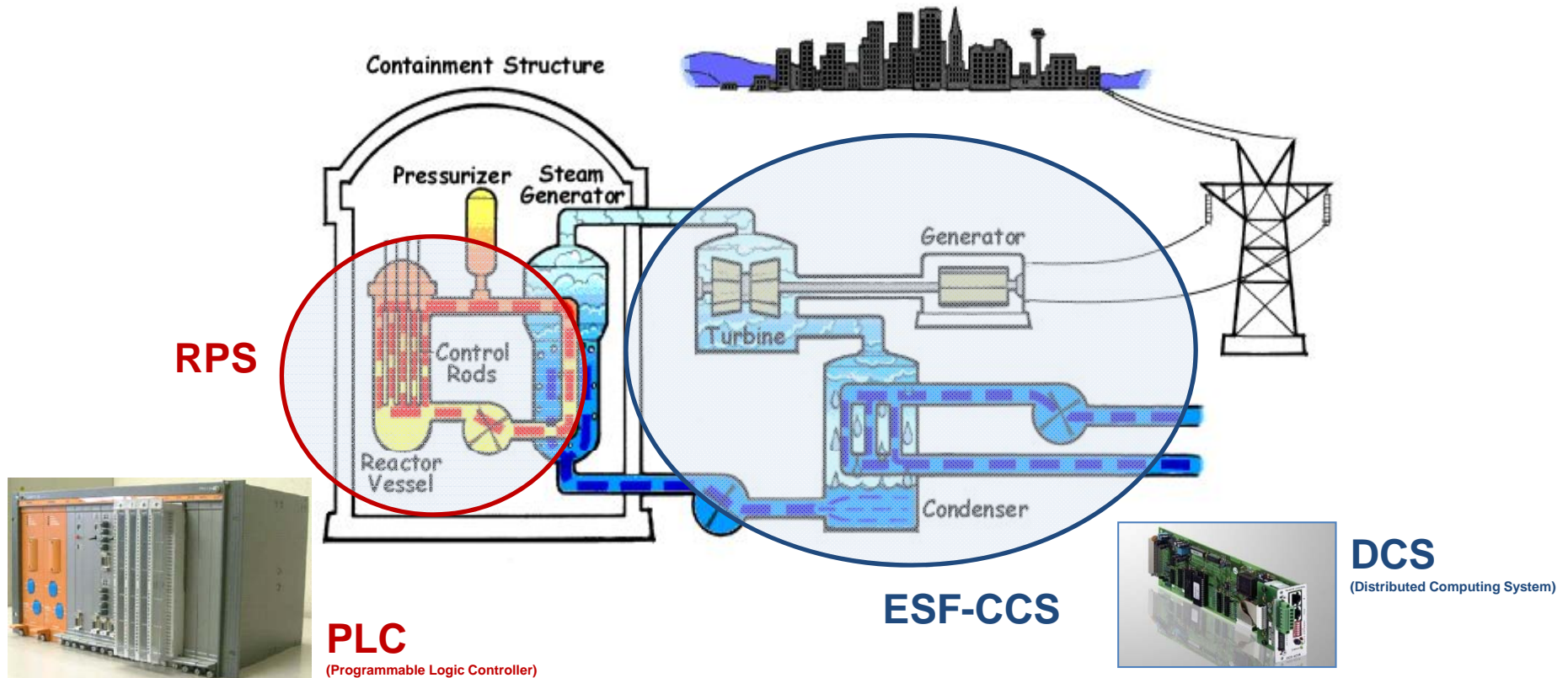
- Introduction
 - Safety-Critical Software in Nuclear Power Plants
 - Software Development Process (Existing vs. Proposed)
- Software Development Process for NPPs
 - Development Process
 - Verification Process
 - Safety Analysis Process
- Conclusion and Future Work

Introduction

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

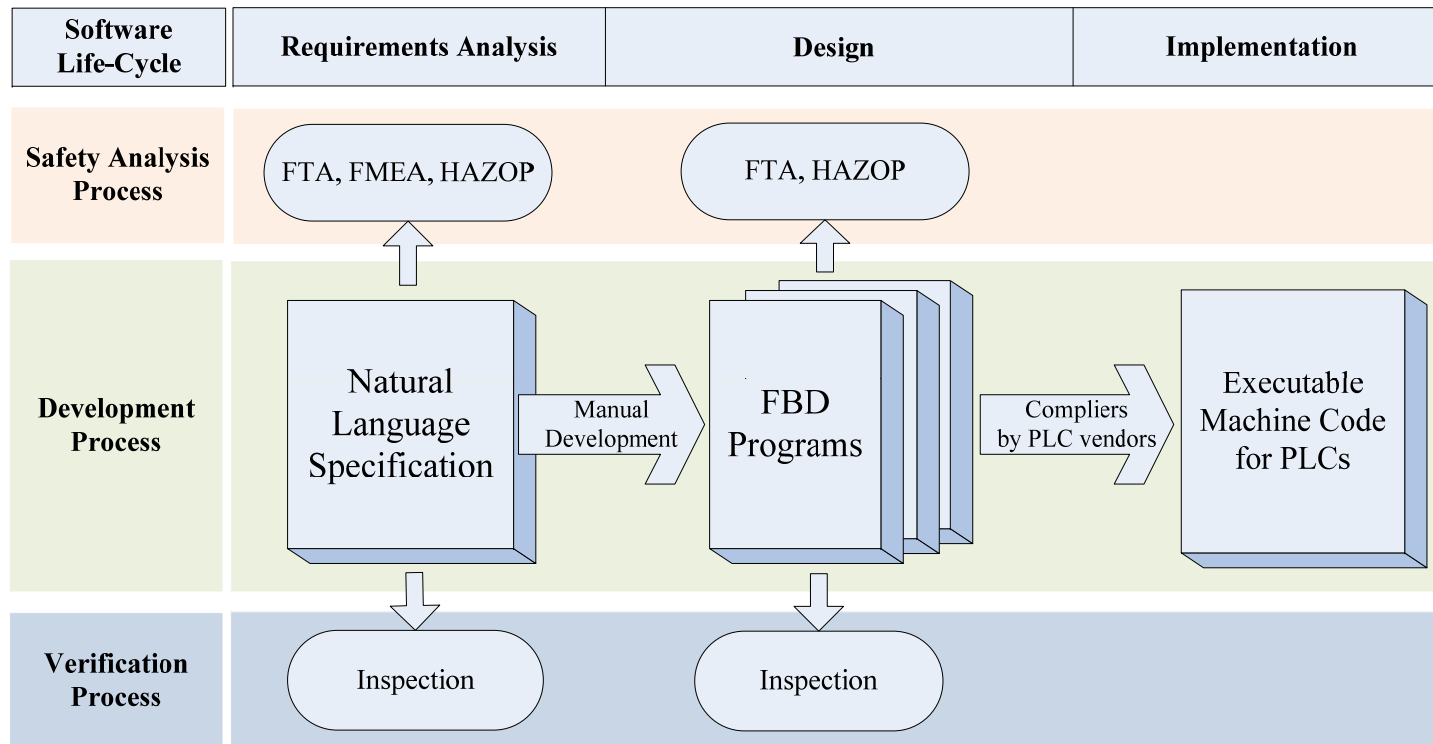
Safety-Critical Software in Nuclear Power Plants

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



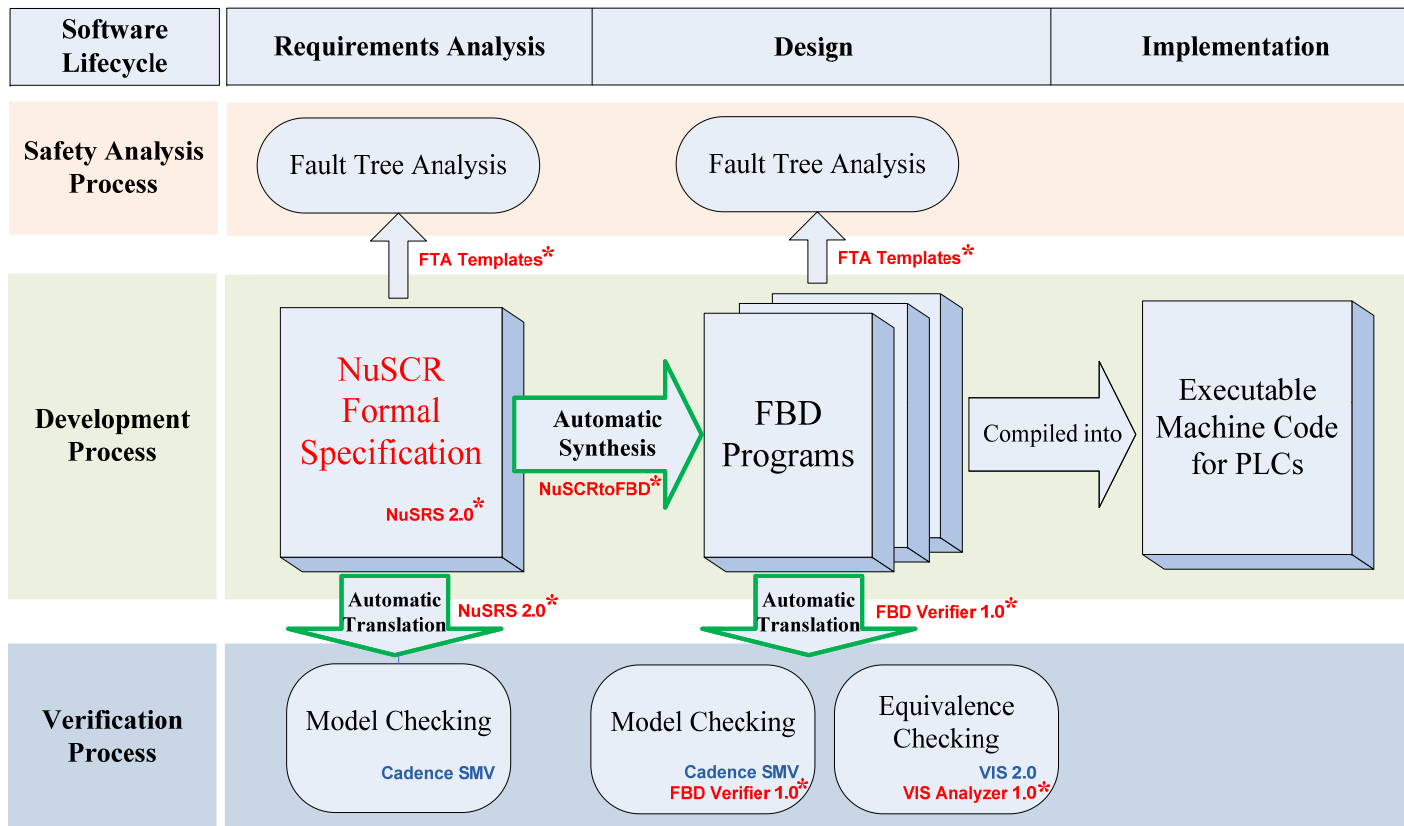
Existing Software Development Process

- 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



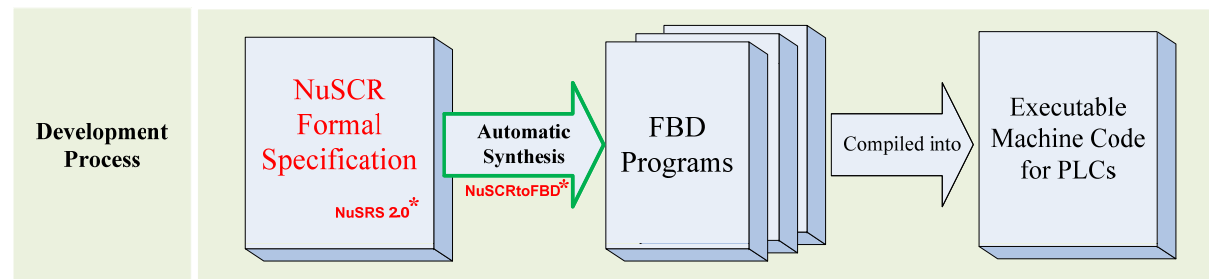
* : Our group's effort

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



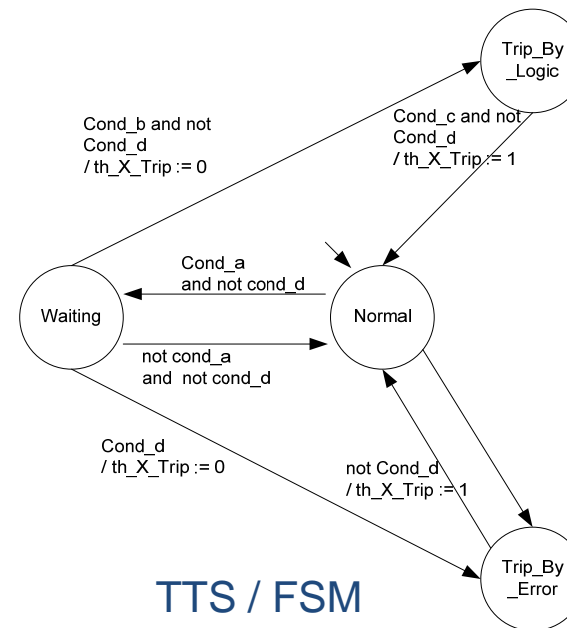
1. Formal Requirements Specification

- 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

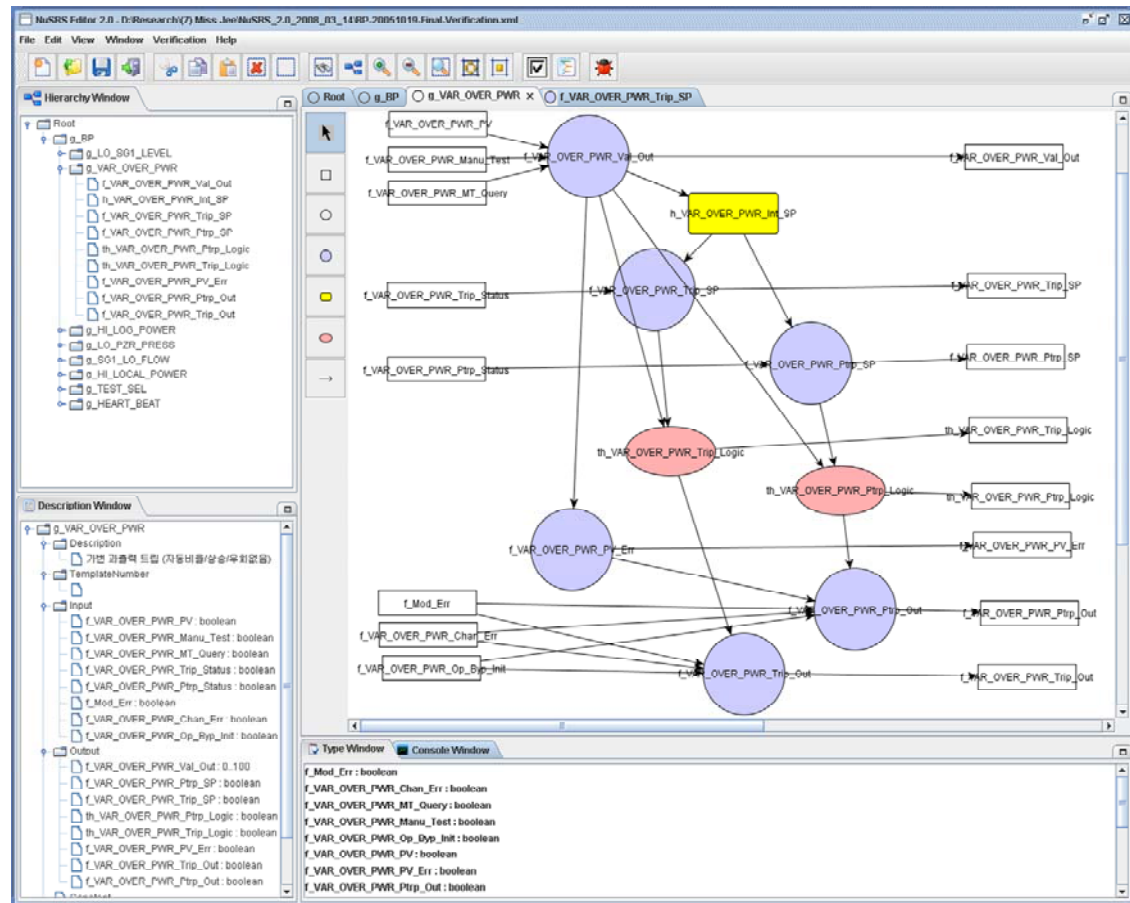
SDT



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)

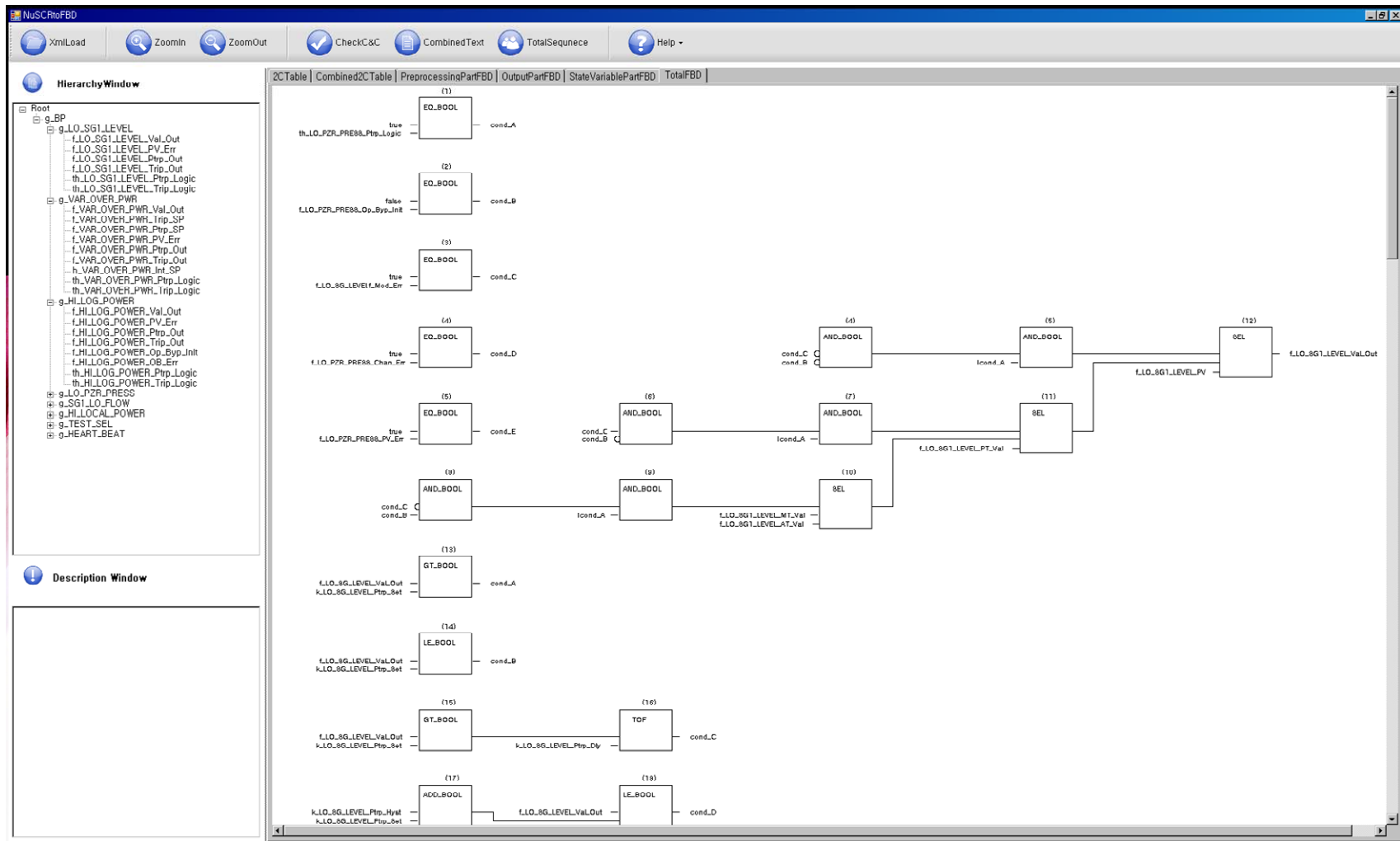


NuSRS (ver. 2.0)

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

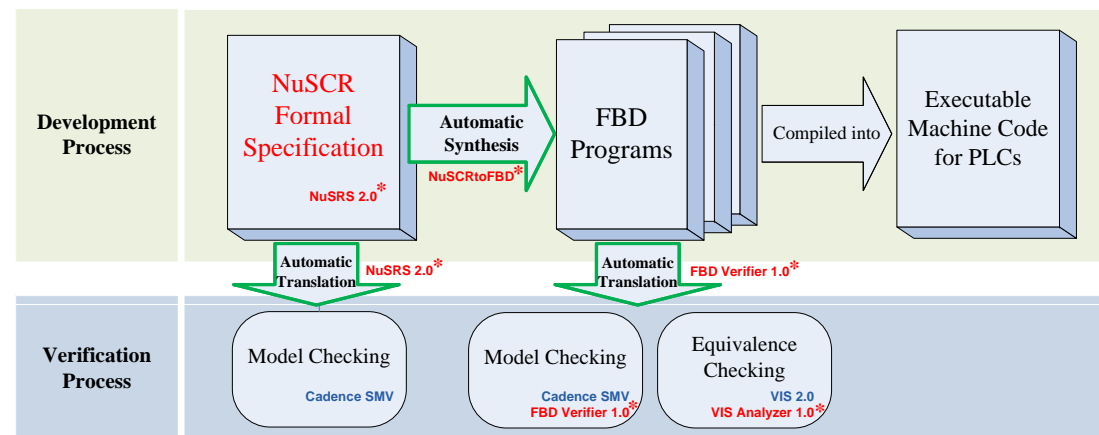


NuSCRtoFBD (ver. 1.0)

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

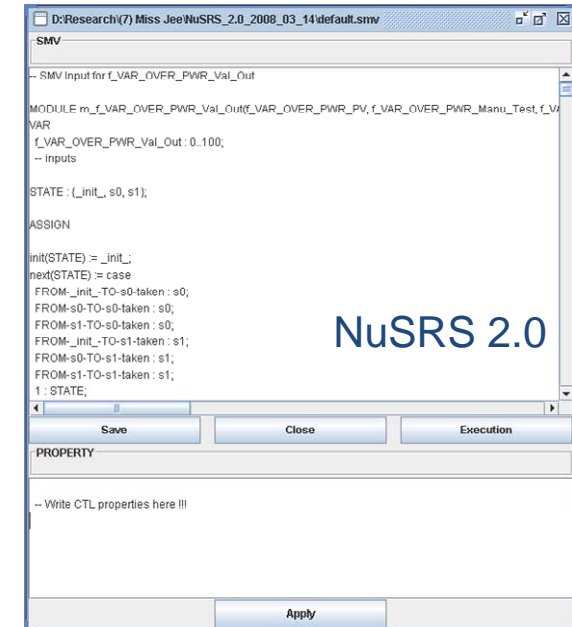
Verification Process

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)



```
-- 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 : 0..100;
-- Inputs

STATE : (_init_, s0, s1);

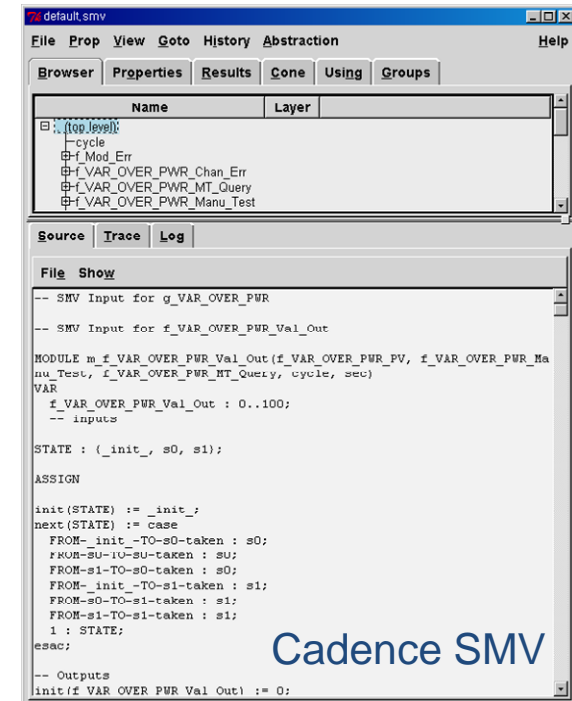
ASSIGN

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 !!!

PROPERTY

Apply
```



```
File Show
-- 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 : 0..100;
-- Inputs

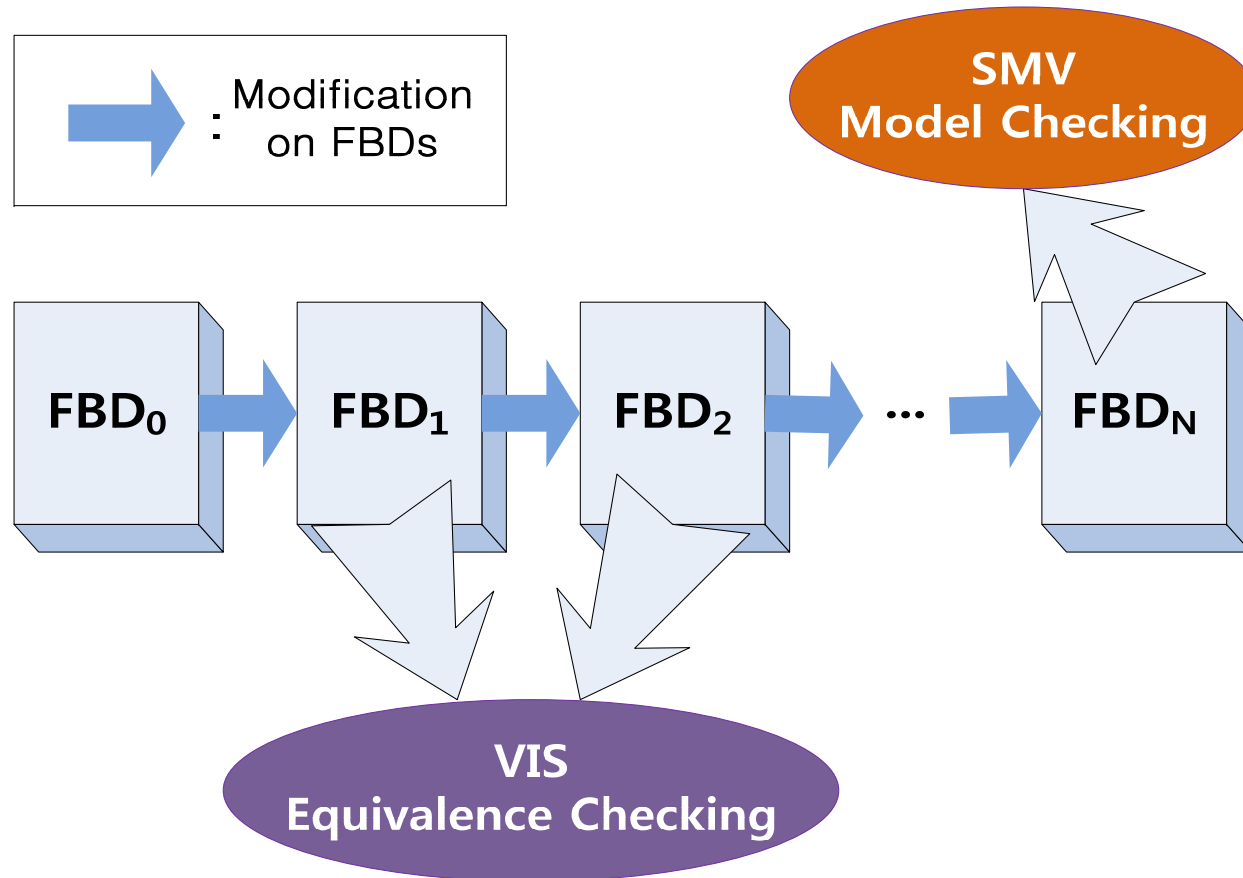
STATE : (_init_, s0, s1);

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

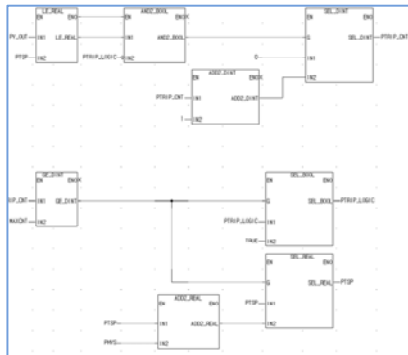
- Formal verification for design specification
 - Target : FBD program
 - Tool : Cadence SMV [5]
 - Technique : LTL model checking

- 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

```

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

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

4. Analyze verification result

3. Perform SMV model checking

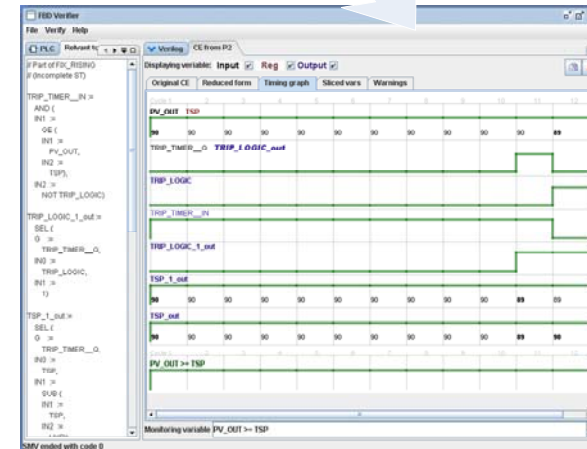
```

//constants
define HYS 0
define PHYS 0
define PTSP_E 0
define TSP_E 0

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

wire TRIP_LOGIC_1;
wire [7:0] TSP_1;
wire TRIP_LOGIC_2;
wire [7:0] TSP_2;
wire TRIP_LOGIC_3;
wire [7:0] TSP_3;
output [7:0] TRIP_LOGIC_out;
output [7:0] TSP_out;
  
```

Cadence SMV



Counterexample viewer in FBD Verifier 1.0

3. Equivalence Checking Designs

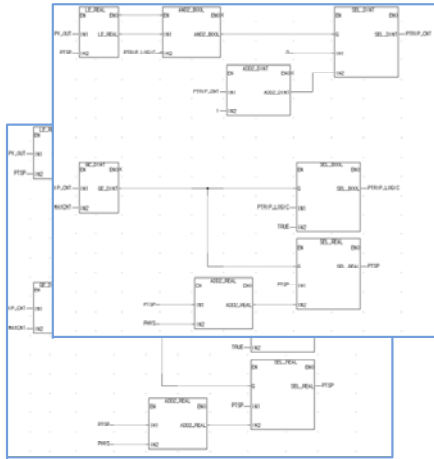
- Formal verification for design specifications
 - Target : Two FBD programs
 - Tool : VIS Verification System [4]
 - Technique : Equivalence checking, Simulation

- 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,
      INC :=
        HV0)
)
PTRIP_LOGIC :=
  SEL (
    G :=
      AND (
        IN1 :=
          LT (
            IN1 :=
              PV_OUT,
              INC :=
                PTP),
            IN :=
              PTRIP_LOGIC,
            IN1 :=
              0)
          )
    PTP :=
      SEL (
        G :=

```

2. Translate into
Verilog programs

3. Read two Verilog programs

VIS Analyzer 1.0 (Source)

```

Verilog 1
C:\KC2007\Te\home\MSVis-2.0\examples\RPS\FBD_Verifier\th_X_Pretrip_Manual.v
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_Dretrip_Rampain 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_Drav_X_Dretrip;
timer_state reg timer;

initial state = S1;
initial th_Drav_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 <

```

4. Perform VIS
equivalence checking

VIS Analyzer 1.0 (Result)

```

Automatic Vis Equivalence Checker
Verilog sources Result Result Table
C:\KC2007\Te\home\MSVis-2.0\examples\RPS\FBD_Verifier\th_X_Pretrip_Manual.v
C:\KC2007\Te\home\MSVis-2.0\examples\RPS\FBD_Verifier\th_X_Pretrip_Mech.v
Vis@ Vis release 2.0 (compiled sat Jun 14 12:02:36 2008)
vis@ vis> vis@ --State 0:
state:NTK2:S1
state:S0
th_Prev_X_Pretrip:NTK2:1
th_Prev_X_Pretrip:1
timer:NTK2:T0
timer:T0

--Goes to state 1:
state:S1
timer:NTK2:T1
timer:T1
--On inputs:
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:NTK2:T2
timer:T2
--On inputs:
<Unchanged>

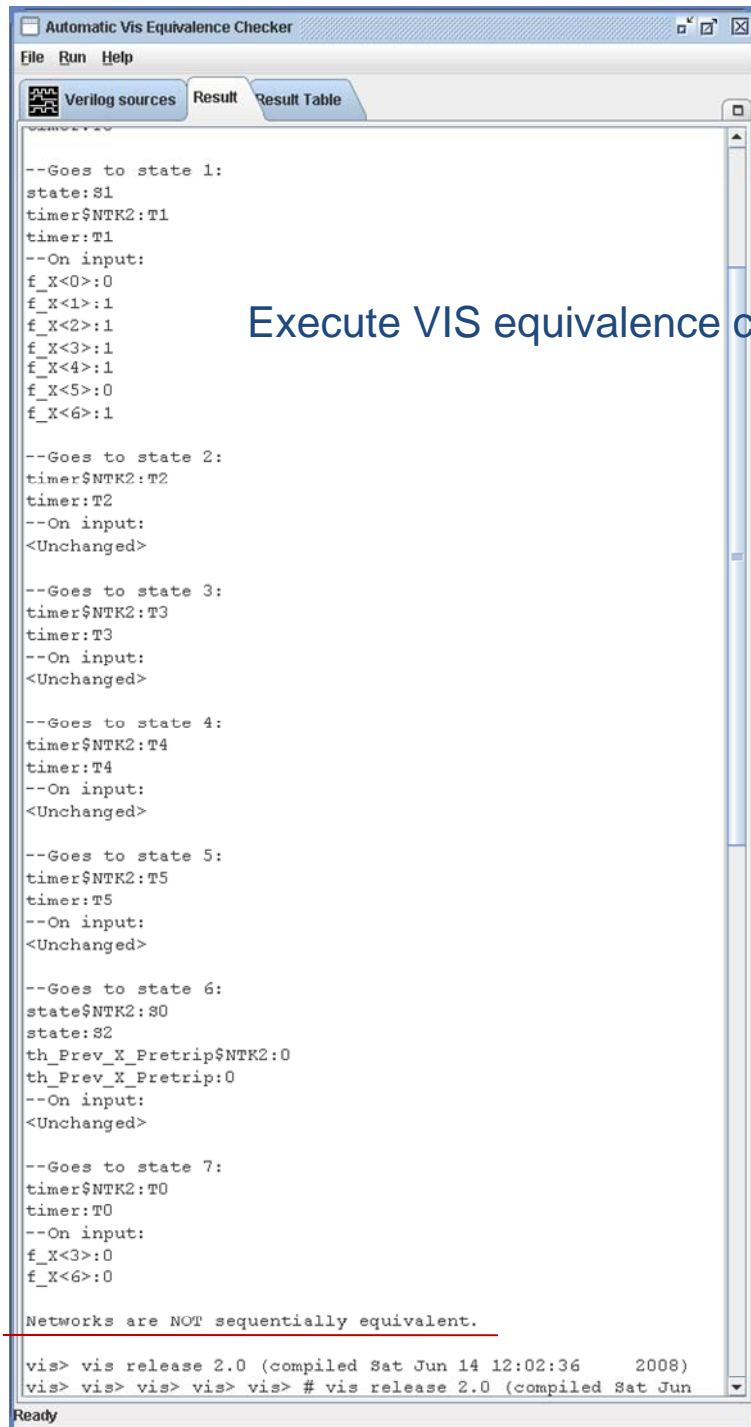
--Goes to state 3:
timer:NTK2:T3
timer:T3
--On inputs:
<Unchanged>

--Goes to state 4:
timer:NTK2:T4
timer:T4

```

5. Verification result with
simulation

Execute VIS equivalence checking



```
Automatic VIS Equivalence Checker
File Run Help
Verilog sources Result Result Table

--Goes to state 1:
state:S1
timer$NTRK2: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$NTRK2:T2
timer:T2
--On input:
<Unchanged>

--Goes to state 3:
timer$NTRK2:T3
timer:T3
--On input:
<Unchanged>

--Goes to state 4:
timer$NTRK2:T4
timer:T4
--On input:
<Unchanged>

--Goes to state 5:
timer$NTRK2:T5
timer:T5
--On input:
<Unchanged>

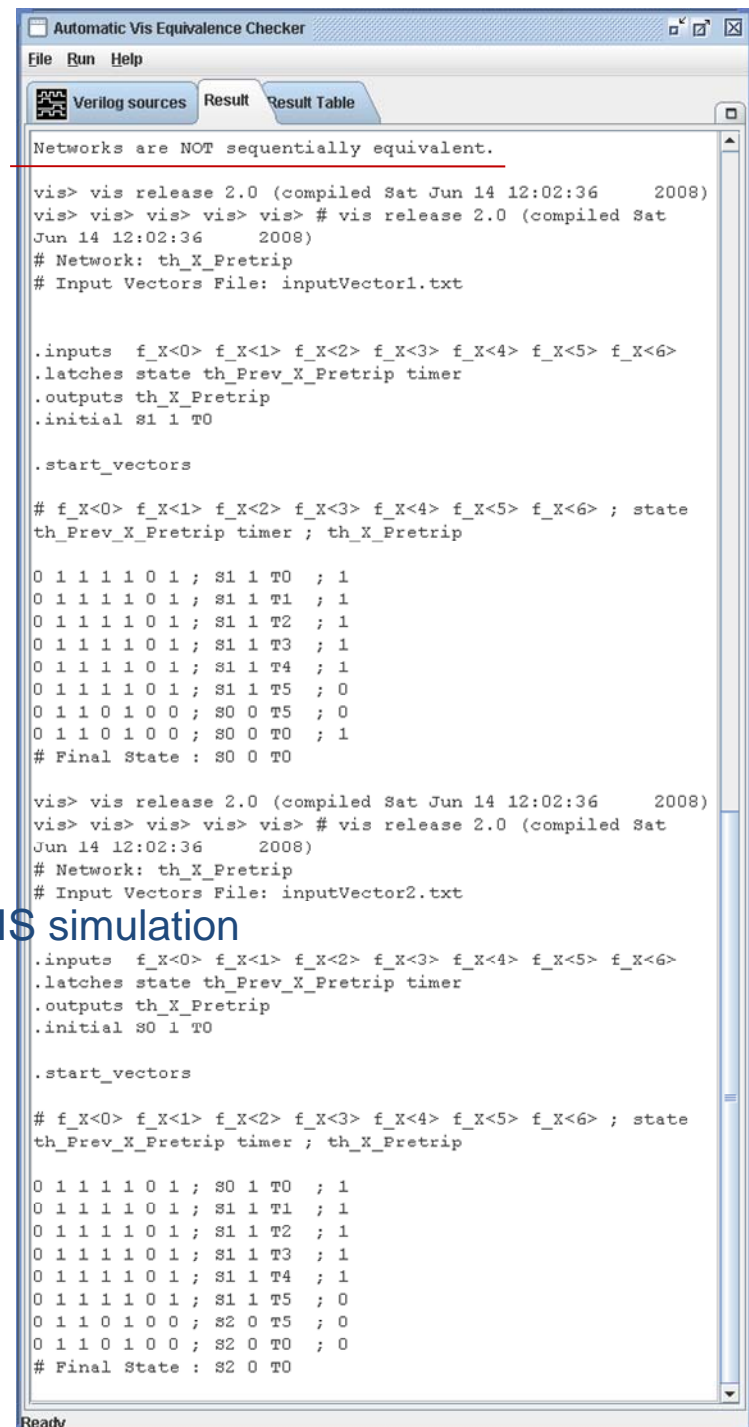
--Goes to state 6:
state$NTRK2:S0
state:S2
th_Prev_X_Pretrip$NTRK2:0
th_Prev_X_Pretrip:0
--On input:
<Unchanged>

--Goes to state 7:
timer$NTRK2:T0
timer:T0
--On input:
f_X<3>:0
f_X<6>:0

Networks are NOT sequentially equivalent.

vis> vis release 2.0 (compiled Sat Jun 14 12:02:36 2008)
vis> vis> vis> vis> vis> # vis release 2.0 (compiled Sat Jun 14 12:02:36 2008)
Ready
```

Execute VIS simulation



```
Automatic VIS Equivalence Checker
File Run Help
Verilog sources Result Result Table

Networks are NOT sequentially equivalent.

vis> vis release 2.0 (compiled Sat Jun 14 12:02:36 2008)
vis> vis> vis> vis> vis> # vis release 2.0 (compiled Sat Jun 14 12:02:36 2008)
# Network: th_X_Pretrip
# Input Vectors File: inputVector1.txt

.inputs f_X<0> f_X<1> f_X<2> f_X<3> f_X<4> f_X<5> f_X<6>
.latches state th_Prev_X_Pretrip timer
.outputs th_X_Pretrip
.initial s1 1 T0

.start_vectors

# f_X<0> f_X<1> f_X<2> f_X<3> f_X<4> f_X<5> f_X<6> ; state
th_Prev_X_Pretrip timer ; th_X_Pretrip

0 1 1 1 1 0 1 ; s1 1 T0 ; 1
0 1 1 1 1 0 1 ; s1 1 T1 ; 1
0 1 1 1 1 0 1 ; s1 1 T2 ; 1
0 1 1 1 1 0 1 ; s1 1 T3 ; 1
0 1 1 1 1 0 1 ; s1 1 T4 ; 1
0 1 1 1 1 0 1 ; s1 1 T5 ; 0
0 1 1 0 1 0 0 ; s0 0 T5 ; 0
0 1 1 0 1 0 0 ; s0 0 T0 ; 1
# Final State : s0 0 T0

vis> vis release 2.0 (compiled Sat Jun 14 12:02:36 2008)
vis> vis> vis> vis> vis> # vis release 2.0 (compiled Sat Jun 14 12:02:36 2008)
# Network: th_X_Pretrip
# Input Vectors File: inputVector2.txt

.inputs f_X<0> f_X<1> f_X<2> f_X<3> f_X<4> f_X<5> f_X<6>
.latches state th_Prev_X_Pretrip timer
.outputs th_X_Pretrip
.initial s0 1 T0

.start_vectors

# f_X<0> f_X<1> f_X<2> f_X<3> f_X<4> f_X<5> f_X<6> ; state
th_Prev_X_Pretrip timer ; th_X_Pretrip

0 1 1 1 1 0 1 ; s0 1 T0 ; 1
0 1 1 1 1 0 1 ; s1 1 T1 ; 1
0 1 1 1 1 0 1 ; s1 1 T2 ; 1
0 1 1 1 1 0 1 ; s1 1 T3 ; 1
0 1 1 1 1 0 1 ; s1 1 T4 ; 1
0 1 1 1 1 0 1 ; s1 1 T5 ; 0
0 1 1 0 1 0 0 ; s2 0 T5 ; 0
0 1 1 0 1 0 0 ; s2 0 T0 ; 0
# Final State : s2 0 T0

Ready
```

Automatic Vis Equivalence Checker

File Run Help

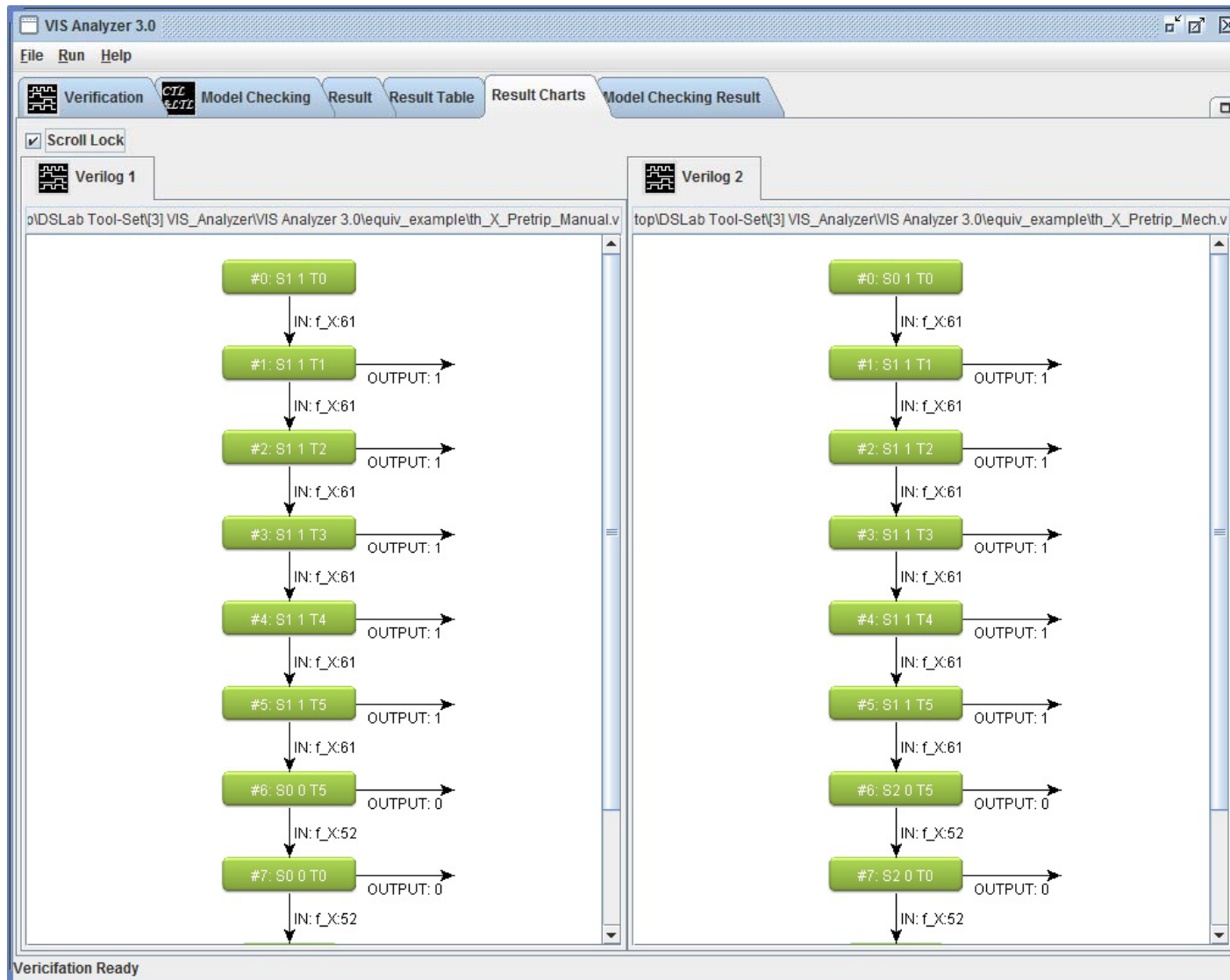
Verilog sources Result Result Table

# 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

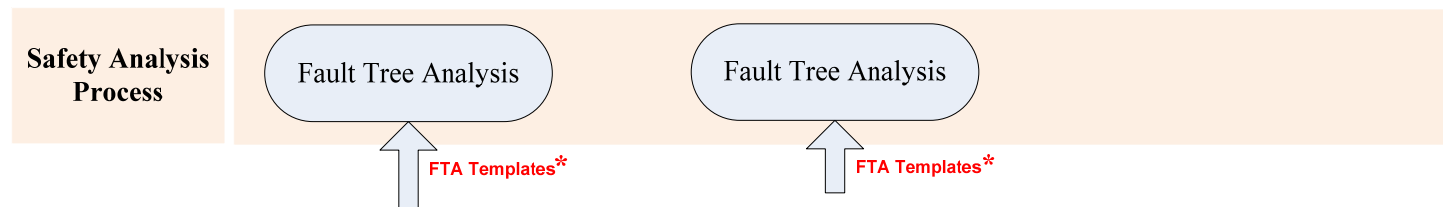


VIS Analyzer (ver. 3.0)

- Visualized and reorganized result - counterexample

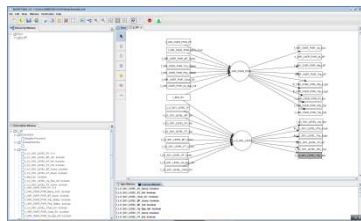
Safety Analysis Process

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



1. Fault Tree Analysis for Requirements

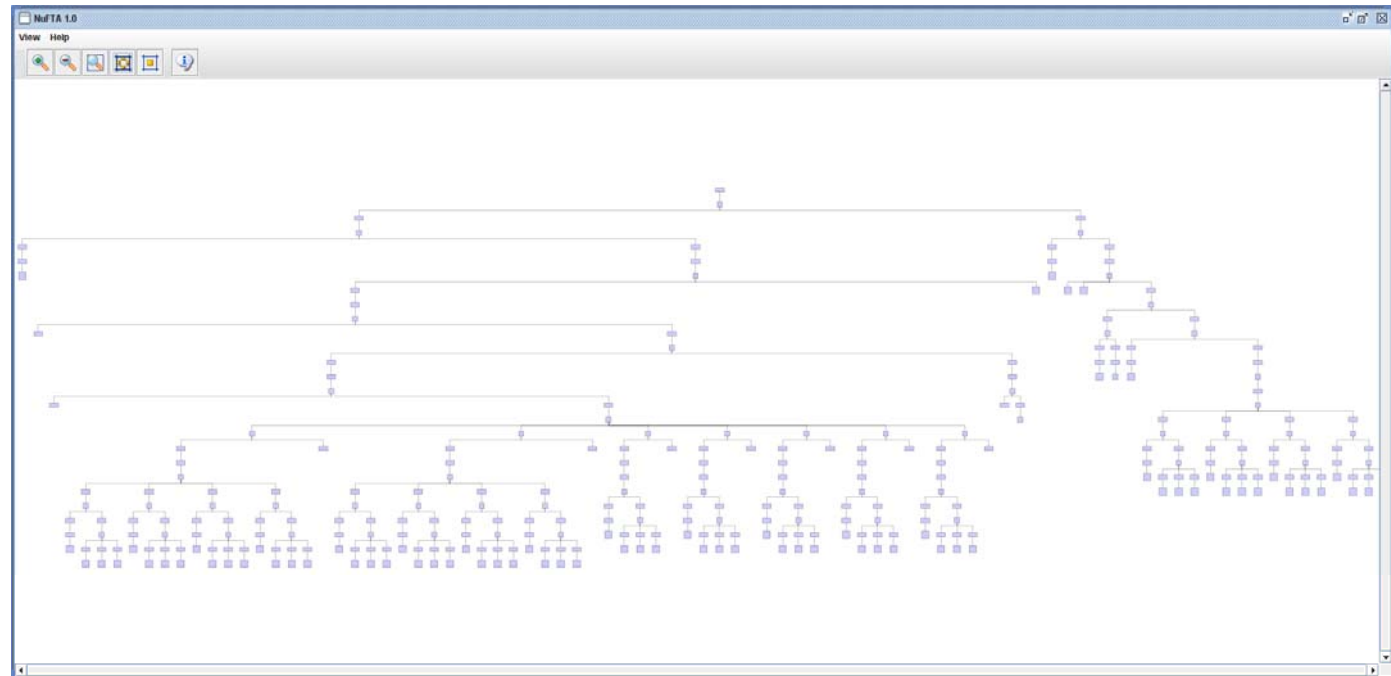
- Fault Tree Analysis
 - Performed manually
 - Totally depends on analyst's experience and ability
- We provided FTA templates and CASE tool (NuFTA) for NuSCR [13]



NuSRS



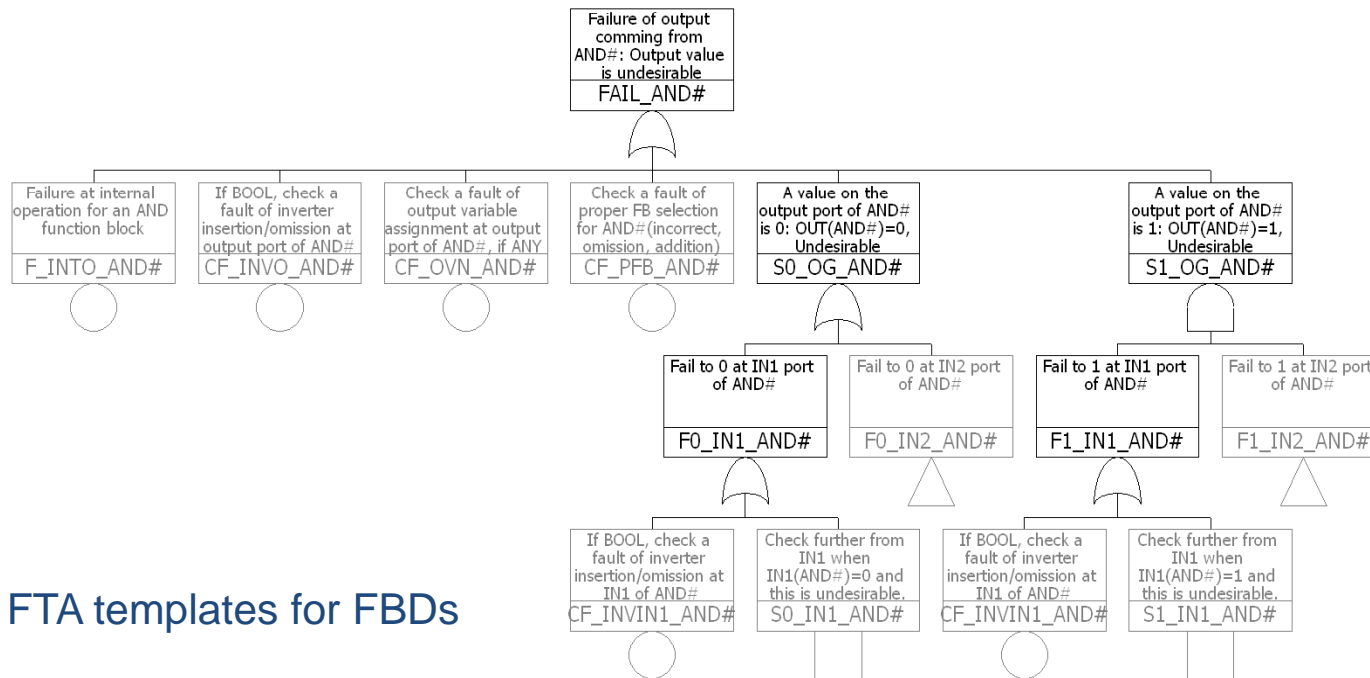
Mechanical
Generation



NuFTA

2. Fault Tree Analysis for Design

- We provided FTA templates and FBD [15]

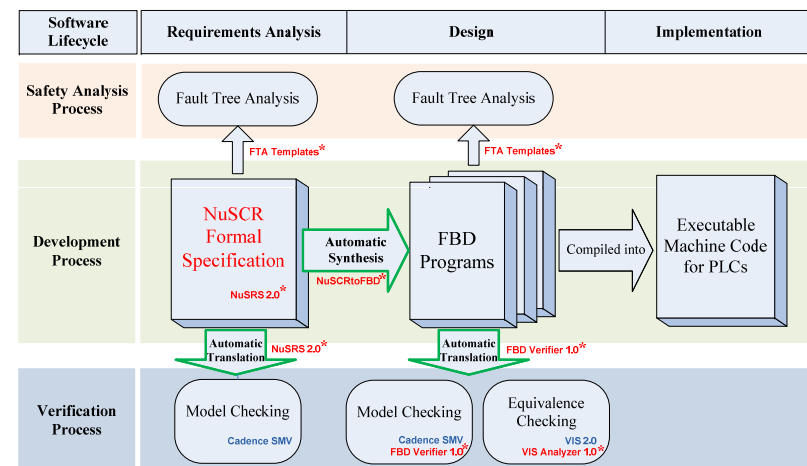


FTA templates for FBDs

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



*: Odr 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

정형 요구사항명세 기반 원자력 소프트웨어 개발 방법론

연구기관:
 - 건국대학교
 - Dependable Software 연구실 (<http://dslab.konkuk.ac.kr>)

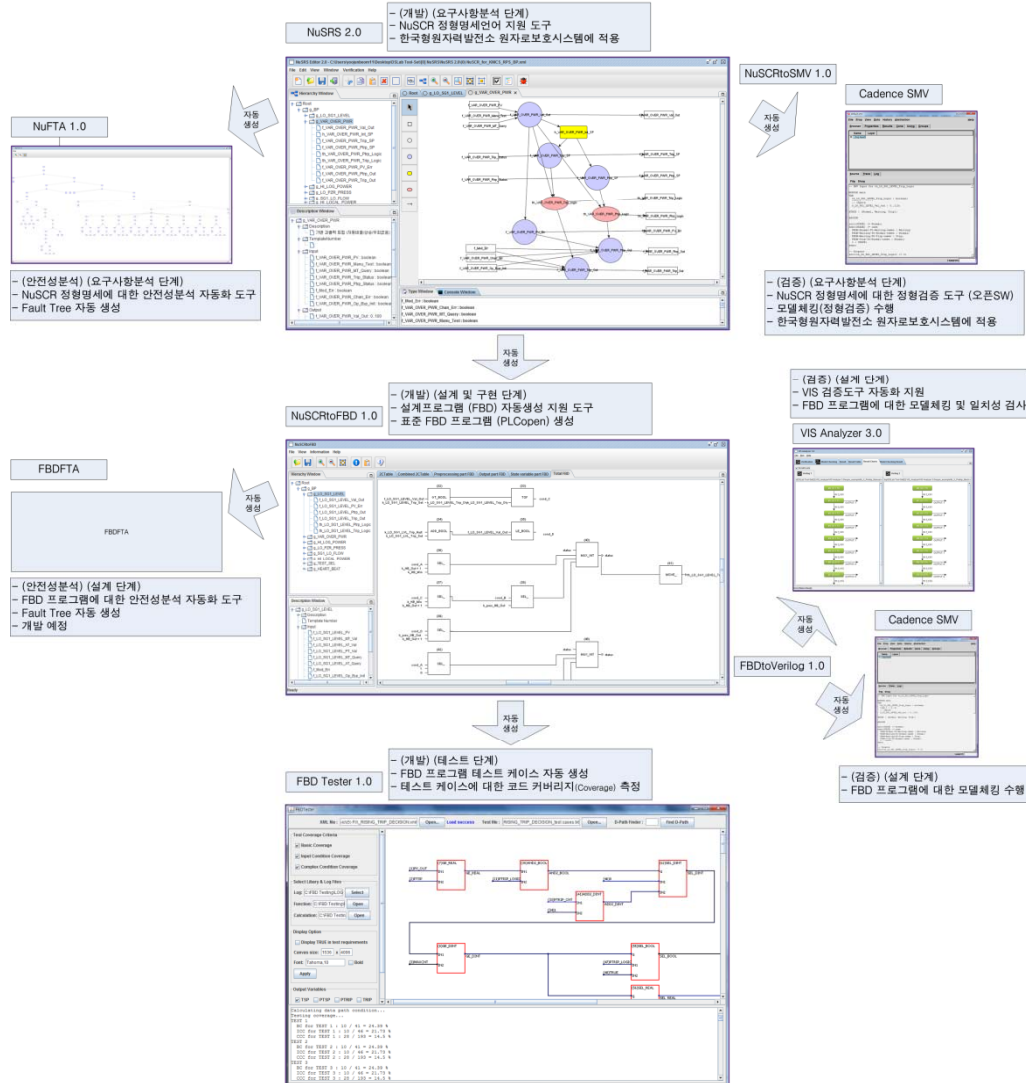
대상 도메인:
 - 원자력발전소 안전최우선 임베디드 시스템용 소프트웨어
 - RPS (Reactor Protection System)

소프트웨어 프로그래밍 언어: FBD (Function Block Diagram)
 임베디드 하드웨어: PLC (Programmable Logic Controller)

지원도구 세트: 총 9종 (7종: 자체개발, 2종: 오픈소스 정형검증 SW)
 개발계획: 통합 Tool-Set 개발 예정 (~2012)

소프트웨어 개발 지원도구 세트:

요구사항 명세 단계	- NuSRS - NuSCRtoSMV - NuFTA - NuSMV	개발 검증 안전성분석 검증
설계 및 구현 단계	- NuSCRtoFBD - FBDFTA - FBDtoVerilog - VIS Analyzer (with VIS)	개발 안전성분석 검증 검증
테스트 단계	- FBD Tester	개발



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.