

NuDE: Development Environment for Safety-Critical Software of Nuclear Power Plant

Jong-Hoon Lee, Junbeom Yoo

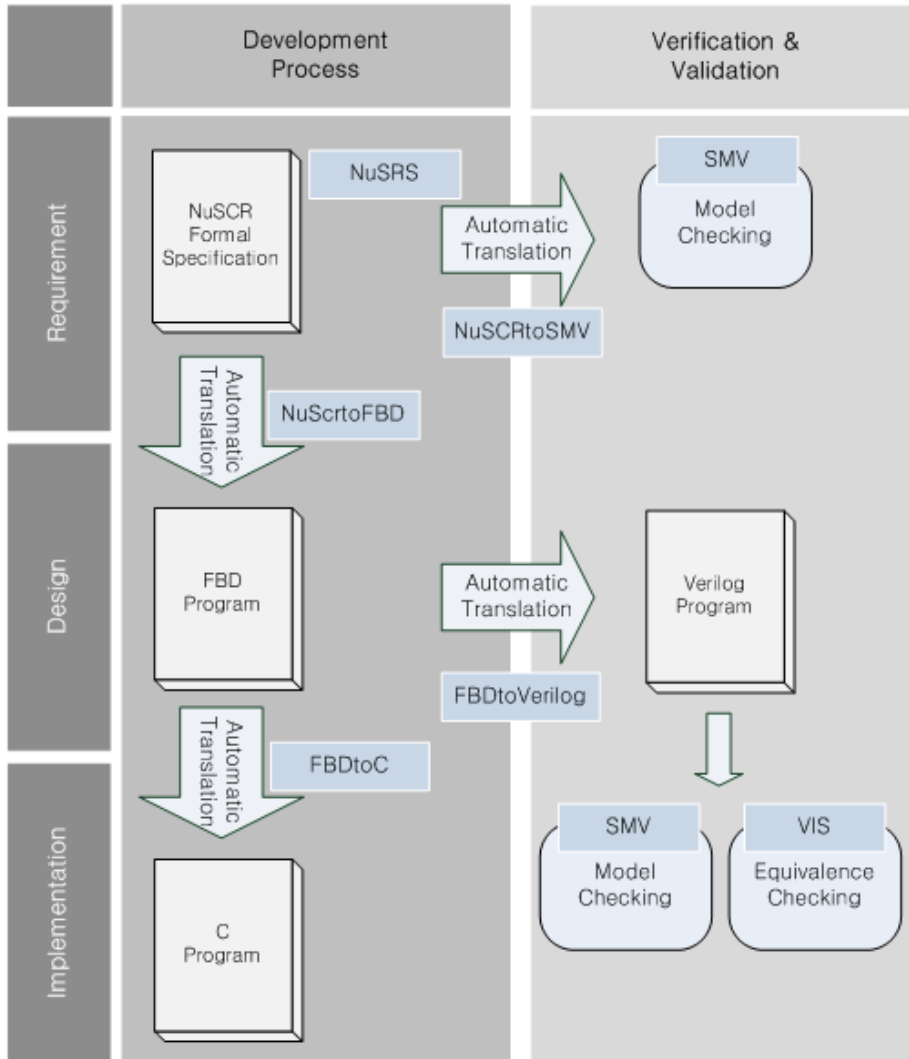
Dependable Software Laboratory

KONKUK University

NuDE: Development Environment for
Safety-Critical Software of Nuclear Power Plant

Overview of NuDE

Development Process in NuDE



Requirements Analysis

- NuSRS
- NuSCRtoSMV(Embedded)

Design Synthesis

- NuSCRtoFBD
- FBDtoVerilog (VIS/SMV)

Implementation

- FBDtoC
- FBDtoVerilog (FPGA/CPLD)

NuDE

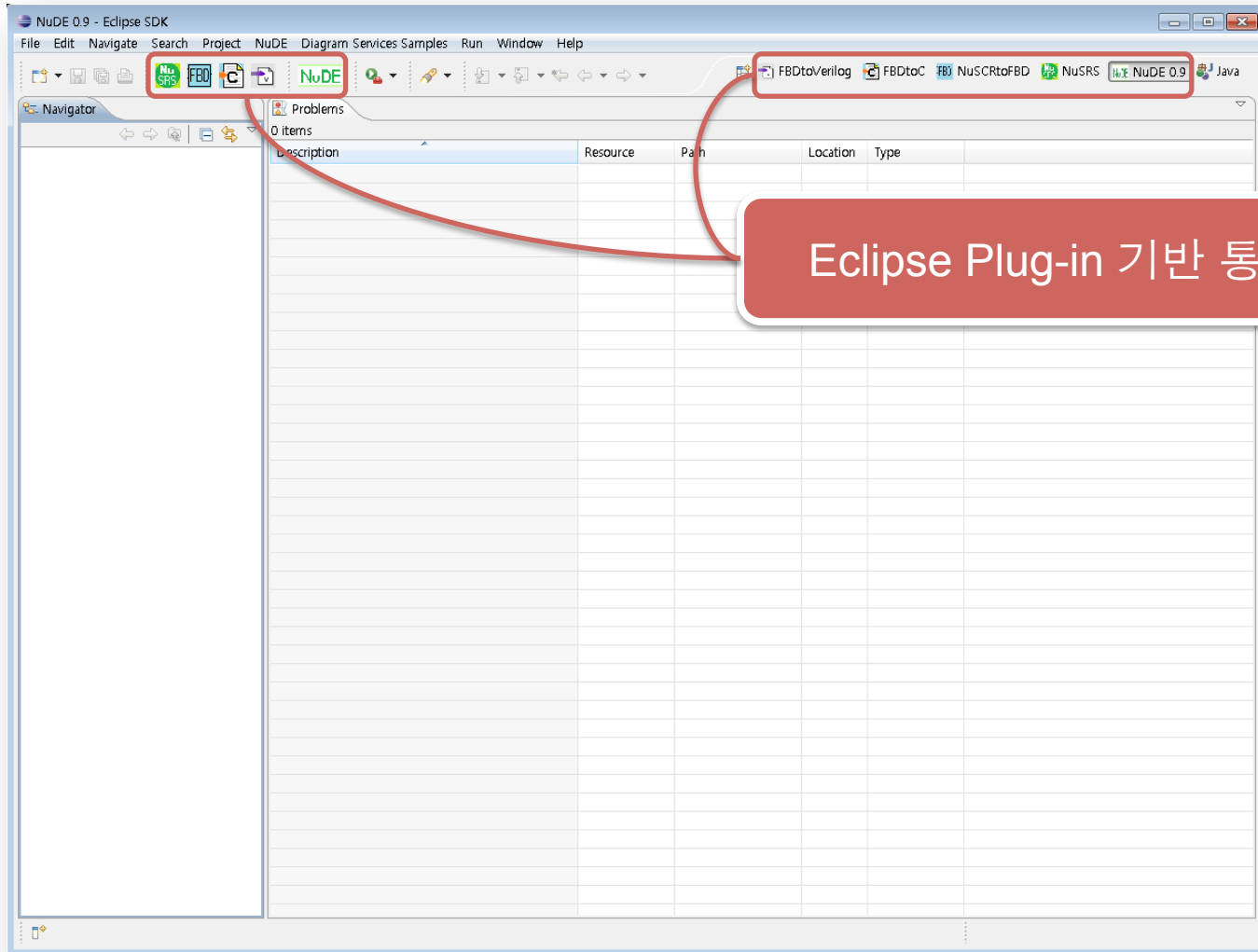
NuDE (Nuclear Development Environment)

- Integration of Existing Tools
 - NuSRS, NuSCRtoFBD, FBDtoVerilog, FBDtoC

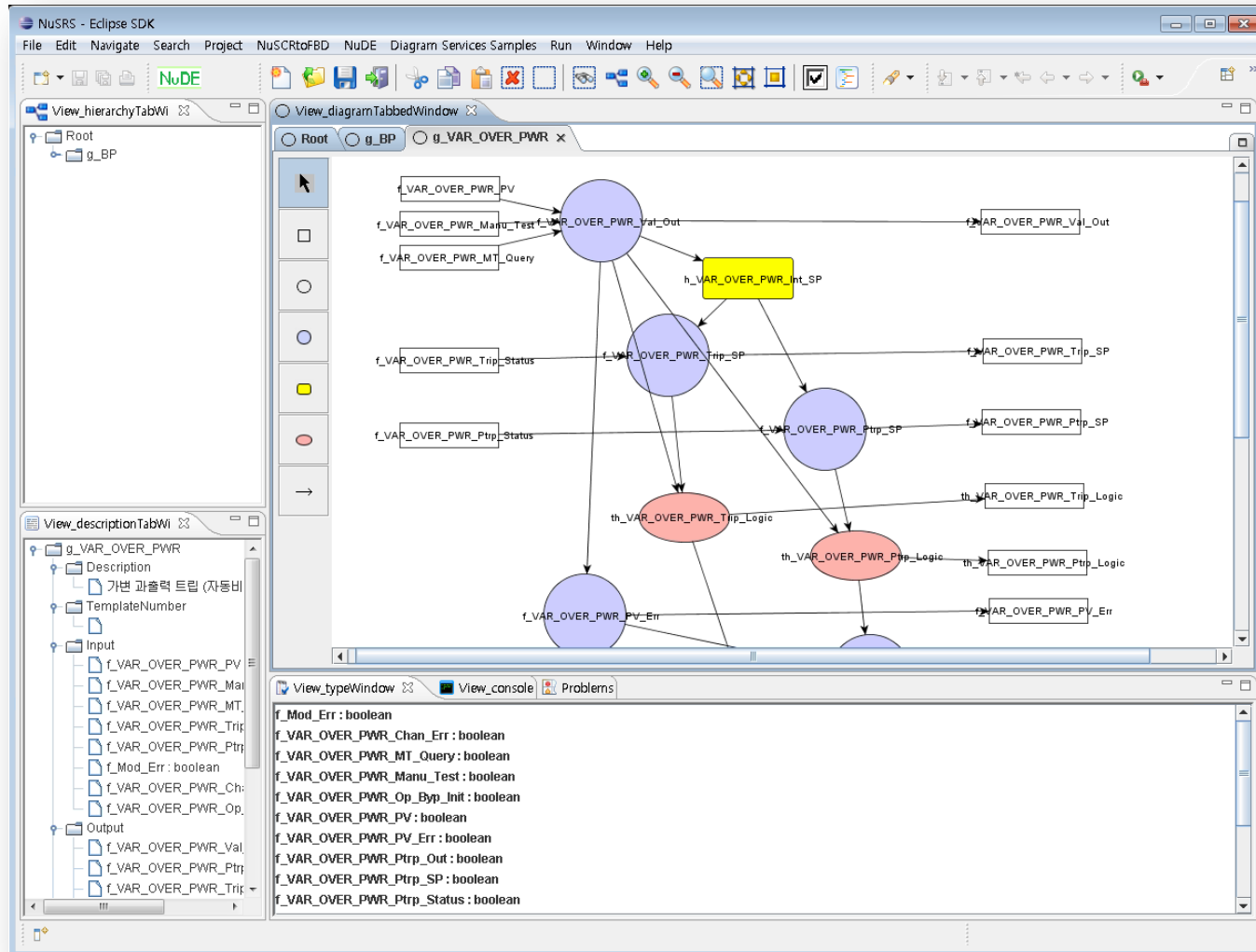
IDE for Nuclear-Domain Software

- Requirement Analysis
 - Formal Requirement Specification (NuSCR)
 - Formal Requirement Verification via SMV
 - SMV Code Generation
- Design Synthesis
 - Automatic Translation from Requirement Specification (FBD)
 - Design Verification via VIS, SMV and HW-CBMC
 - Verilog Code Generation
- Implementation
 - C Code Generation
 - Verilog Code Generation for FPGA/CPLD

NuDE



Requirements Analysis – NuSRS



Requirements Verification – NuSCRtoSMV

The screenshot displays the NuSRS Eclipse SDK environment. The main window shows a diagram with nodes and arrows, representing a state machine or flowchart. The left sidebar contains a 'View_hierarchyTabWi' and a 'View_descriptionTabWi' showing a tree structure of components and their descriptions. The right pane shows the SMV code for the module.

Diagram Description: The diagram shows a central node 'f_VAR_OVER_PWR' with several input nodes: 'f_VAR_OVER_PWR_PV', 'f_VAR_OVER_PWR_Manu_Test', 'f_VAR_OVER_PWR_MT_Query', 'f_VAR_OVER_PWR_Trip_Status', and 'f_VAR_OVER_PWR_Ptrip_Status'. Arrows indicate dependencies or data flow from these inputs to the central node.

SMV Code:

```

-- 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, f_VAR_OVER_PWR_Trip_Status, f_VAR_OVER_PWR_Ptrip_Status, 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-s0-TO-s1-taken : s1;
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;

```

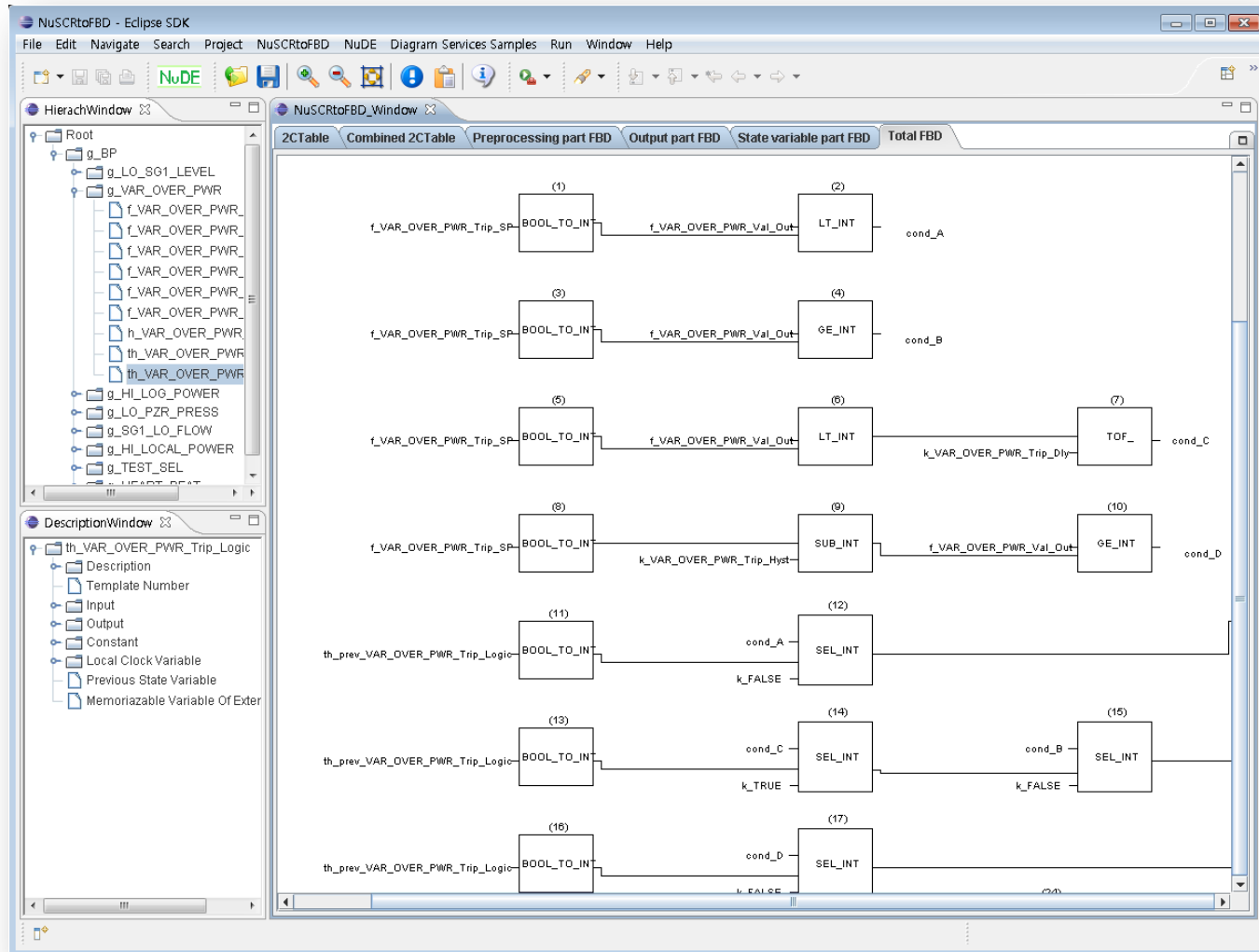
Property List:

```

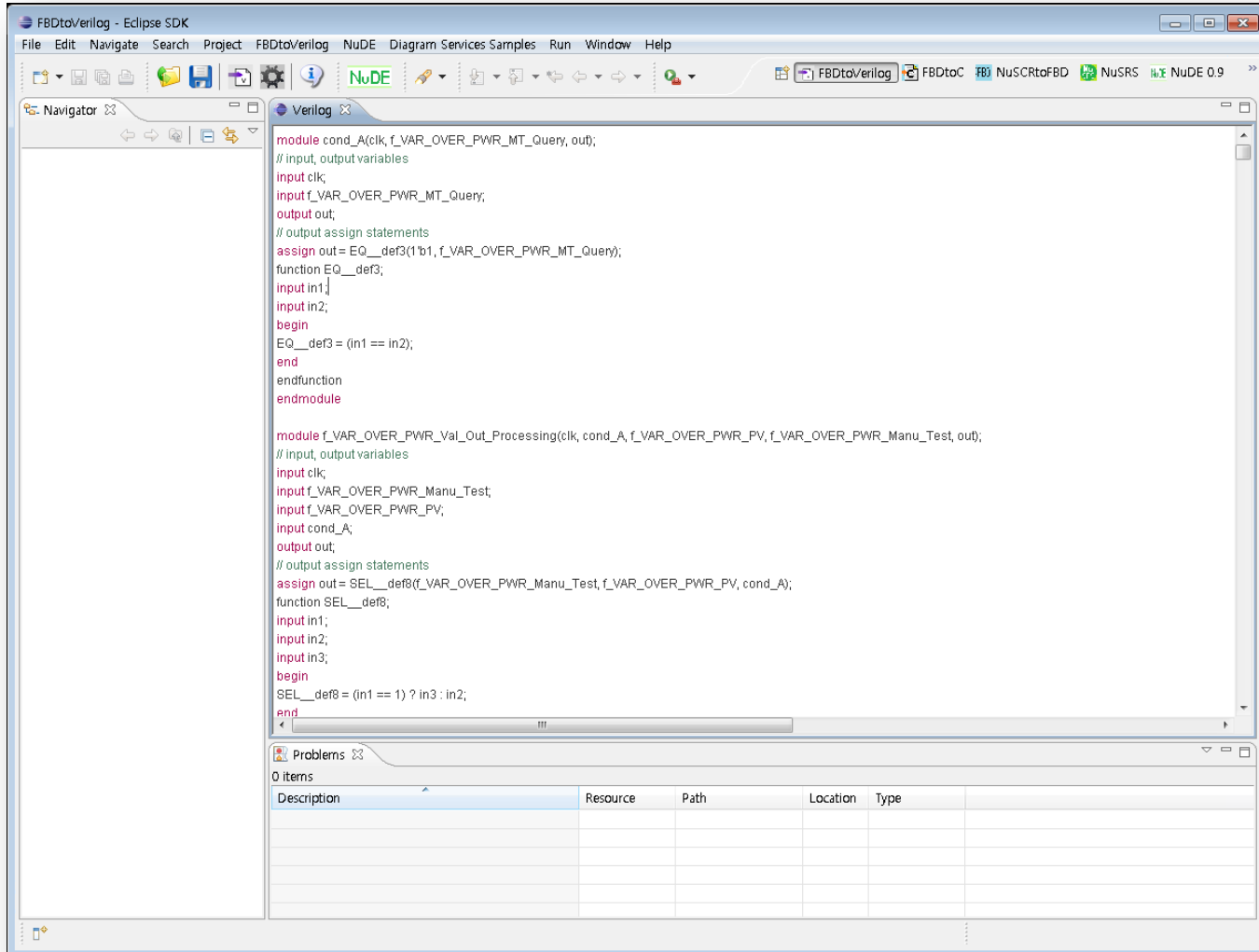
f_Mod_Err : boolean
f_VAR_OVER_PWR_Chan_Err : boolean
f_VAR_OVER_PWR_MT_Query : boolean
f_VAR_OVER_PWR_Manu_Test : boolean
f_VAR_OVER_PWR_Op_Byp_Init : boolean
f_VAR_OVER_PWR_PV : boolean
f_VAR_OVER_PWR_PV_Err : boolean
f_VAR_OVER_PWR_Ptrip_Out : boolean
f_VAR_OVER_PWR_Ptrip_SP : boolean
f_VAR_OVER_PWR_Ptrip_Status : boolean

```

Design Synthesis – NuSCRtoFBD



Design Verification – FBDtoVerilog



The screenshot shows the Eclipse IDE interface for the 'FBDtoVerilog' project. The main editor displays Verilog code with two modules: 'cond_A' and 'f_VAR_OVER_PWR_Val_Out_Processing'. The 'cond_A' module defines an equality function 'EQ__def3'. The 'f_VAR_OVER_PWR_Val_Out_Processing' module defines a selection function 'SEL__def8'. The Problems view at the bottom is empty, indicating no errors.

```

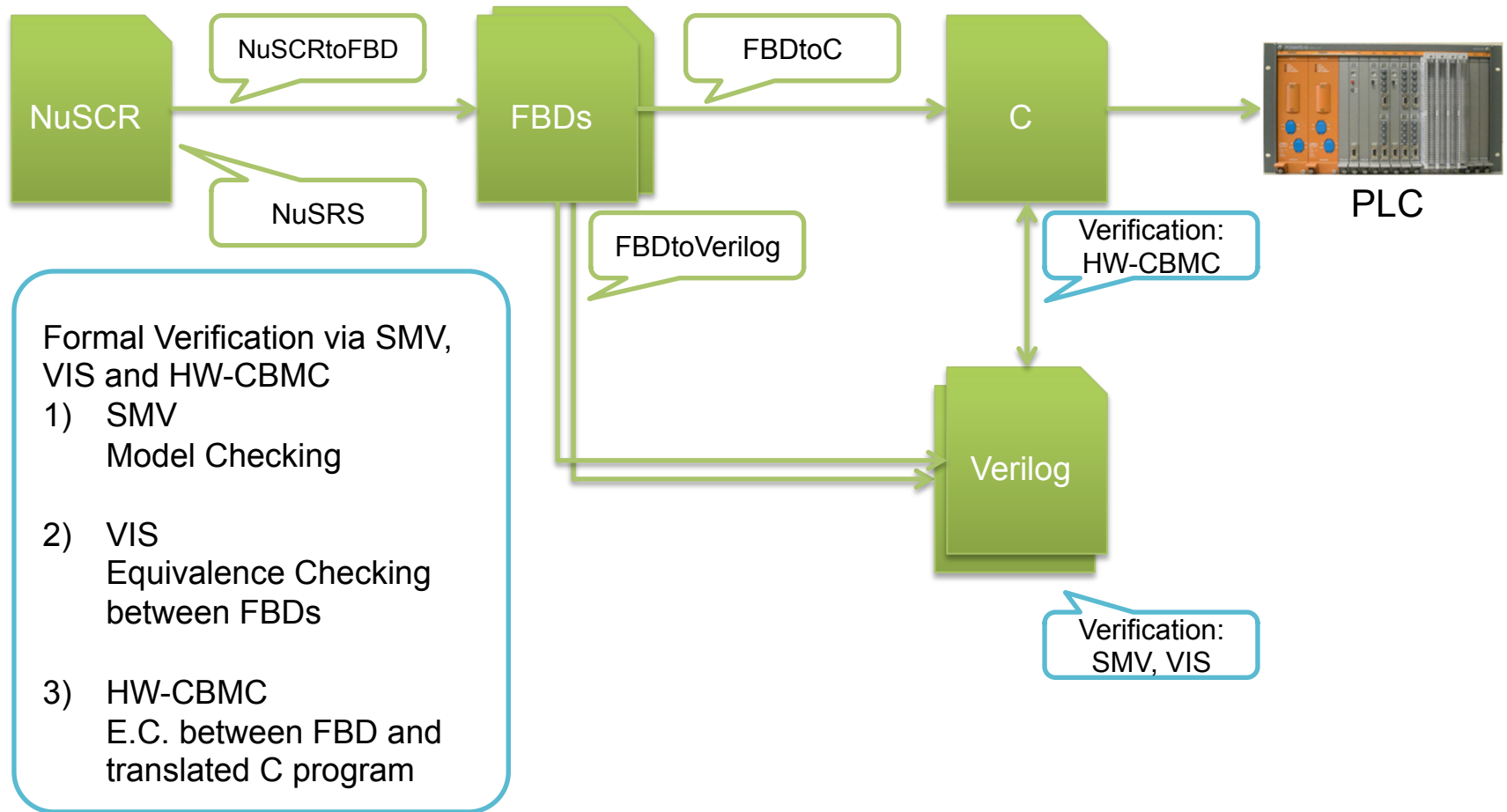
module cond_A(clk, f_VAR_OVER_PWR_MT_Query, out);
// input, output variables
input clk;
input f_VAR_OVER_PWR_MT_Query;
output out;
// output assign statements
assign out = EQ__def3(1'b1, f_VAR_OVER_PWR_MT_Query);
function EQ__def3;
input in1;
input in2;
begin
EQ__def3 = (in1 == in2);
end
endfunction
endmodule

module f_VAR_OVER_PWR_Val_Out_Processing(clk, cond_A, f_VAR_OVER_PWR_PV, f_VAR_OVER_PWR_Manu_Test, out);
// input, output variables
input clk;
input f_VAR_OVER_PWR_Manu_Test;
input f_VAR_OVER_PWR_PV;
input cond_A;
output out;
// output assign statements
assign out = SEL__def8(f_VAR_OVER_PWR_Manu_Test, f_VAR_OVER_PWR_PV, cond_A);
function SEL__def8;
input in1;
input in2;
input in3;
begin
SEL__def8 = (in1 == 1) ? in3 : in2;
end
endfunction
endmodule

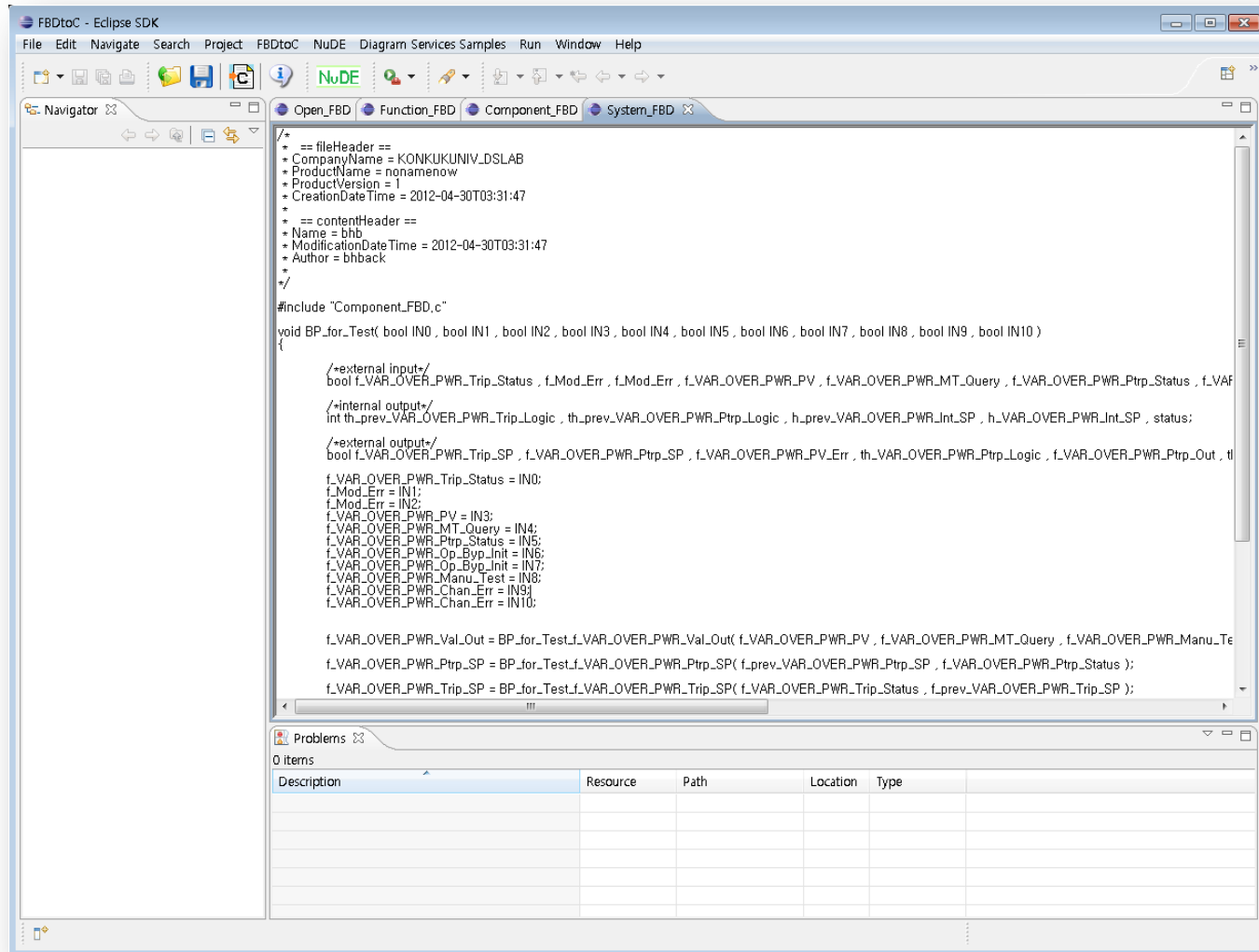
```

| Description | Resource | Path | Location | Type |
|-------------|----------|------|----------|------|
| | | | | |
| | | | | |
| | | | | |
| | | | | |

Design Verification – FBDtoVerilog (Con'td)



Implementation – FBDtoC



```

* == fileHeader ==
* CompanyName = KONKUKUNIV_DSLAB
* ProductName = nonamenow
* ProductVersion = 1
* CreationDateTime = 2012-04-30T03:31:47
*
* == contentHeader ==
* Name = bbb
* ModificationDateTime = 2012-04-30T03:31:47
* Author = bhback
*/
#include "Component_FBD.c"

void BP_for_Test( bool IN0 , bool IN1 , bool IN2 , bool IN3 , bool IN4 , bool IN5 , bool IN6 , bool IN7 , bool IN8 , bool IN9 , bool IN10 )
{
    /*-external input-*/
    bool f_VAR_OVER_PWR_Trip_Status , f_Mod_Err , f_Mod_Err , f_VAR_OVER_PWR_PV , f_VAR_OVER_PWR_MT_Query , f_VAR_OVER_PWR_Ptrp_Status , f_VAR_OVER_PWR_Ptrp_Status ;
    /*-internal output-*/
    int th_prev_VAR_OVER_PWR_Trip_Logic , th_prev_VAR_OVER_PWR_Ptrp_Logic , h_prev_VAR_OVER_PWR_Int_SP , h_VAR_OVER_PWR_Int_SP , status;
    /*-external output-*/
    bool f_VAR_OVER_PWR_Trip_SP , f_VAR_OVER_PWR_Ptrp_SP , f_VAR_OVER_PWR_PV_Err , th_VAR_OVER_PWR_Ptrp_Logic , f_VAR_OVER_PWR_Ptrp_Out , f_VAR_OVER_PWR_Ptrp_Out ;

    f_VAR_OVER_PWR_Trip_Status = IN0;
    f_Mod_Err = IN1;
    f_Mod_Err = IN2;
    f_VAR_OVER_PWR_PV = IN3;
    f_VAR_OVER_PWR_MT_Query = IN4;
    f_VAR_OVER_PWR_Ptrp_Status = IN5;
    f_VAR_OVER_PWR_Op_By_Init = IN6;
    f_VAR_OVER_PWR_Op_By_Init = IN7;
    f_VAR_OVER_PWR_Manu_Test = IN8;
    f_VAR_OVER_PWR_Chan_Err = IN9;
    f_VAR_OVER_PWR_Chan_Err = IN10;

    f_VAR_OVER_PWR_Val_Out = BP_for_Test.f_VAR_OVER_PWR_Val_Out( f_VAR_OVER_PWR_PV , f_VAR_OVER_PWR_MT_Query , f_VAR_OVER_PWR_Manu_Test );
    f_VAR_OVER_PWR_Ptrp_SP = BP_for_Test.f_VAR_OVER_PWR_Ptrp_SP( f_prev_VAR_OVER_PWR_Ptrp_SP , f_VAR_OVER_PWR_Ptrp_Status );
    f_VAR_OVER_PWR_Trip_SP = BP_for_Test.f_VAR_OVER_PWR_Trip_SP( f_VAR_OVER_PWR_Trip_Status , f_prev_VAR_OVER_PWR_Trip_SP );
}
    
```

Considerations for FPGA/CPLD

NPP Software based on PLC

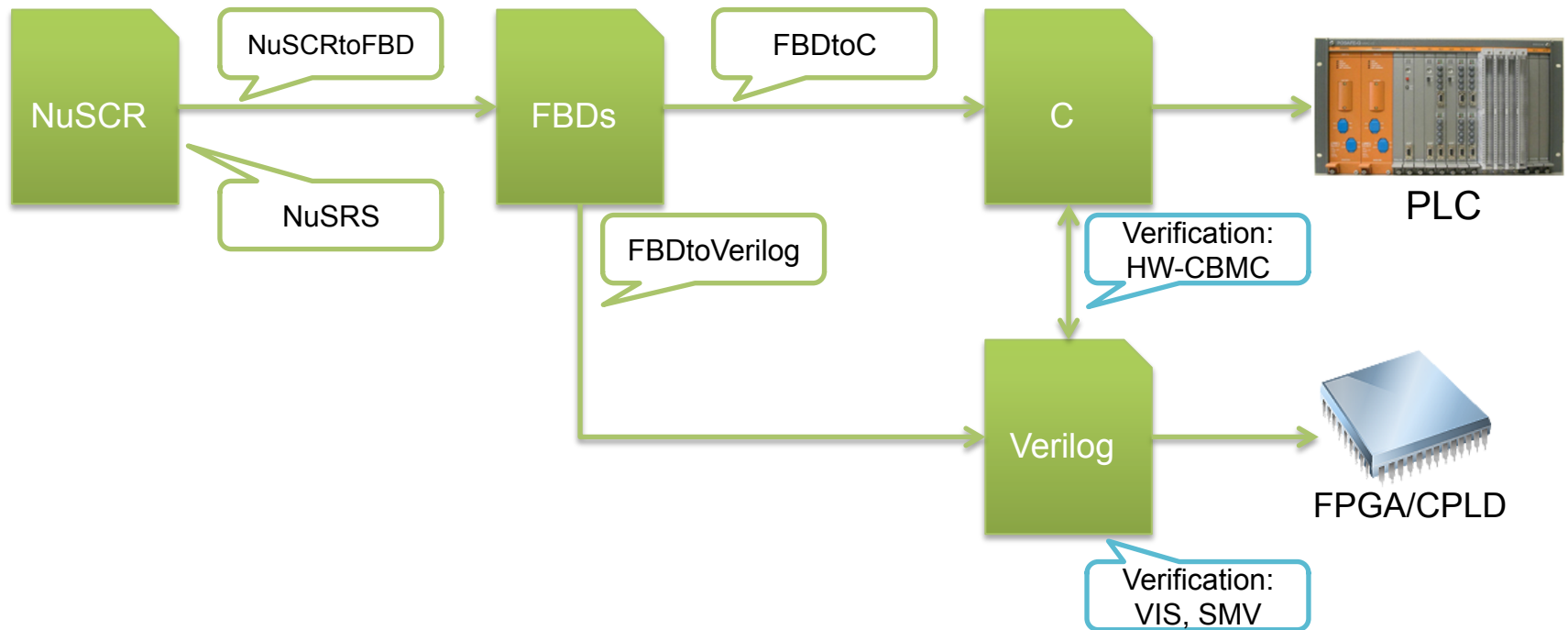
- Implementation: FBD or C Code
 - 기존 PLC 기반 SW에서는 FBD나 C Code를 구현으로 사용



NPP Software based on FPGA/CPLD

- Implementation: Verilog HDL
 - FPGA/CPLD 기반 시스템에 대한 연구들이 진행 중
 - FPGA/CPLD는 Verilog HDL을 구현으로 사용

Considerations for FPGA/CPLD (Cont'd)



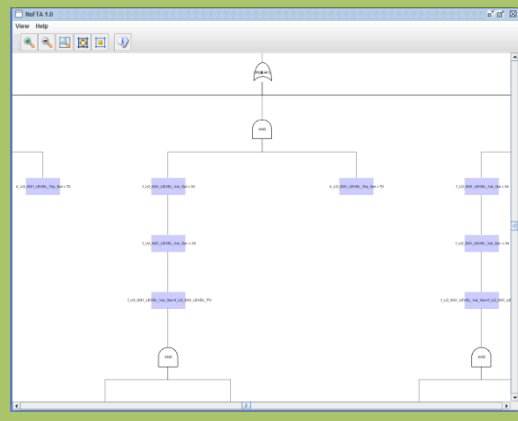
NuDE: Development Environment for
Safety-Critical Software of Nuclear Power Plant

Improvements of NuDE

Not Yet Integrated

NuFTA

- FTA for Requirements Specification



VIS Analyzer

- Automated VIS Equivalence Checking

The screenshot displays two Verilog code snippets in a side-by-side comparison view. The left snippet is labeled 'Verilog 1' and the right is 'Verilog 2'. Both snippets define modules for logic simulation and timing analysis.

```

Verilog 1
C:\Users\Jonghoon\Desktop\FBD\FBD@the_cb_X_Prefig_satu_01.v
in2;
assign OUT = (in1 == 1) ? in3 :
endmodule
module MOVE(in1, OUT);
input in1;
output OUT;
assign OUT = in1;
endmodule
module TOP(clk, IN, DELAY, OUT);
input clk;
input IN;
input [0:6] DELAY;
output OUT;
reg[0:2] timer;
initial timer = 3'b000;
assign OUT = !IN && (timer ==
DELAY) ? 0;
always @(posedge clk) begin
if(!IN) begin

```

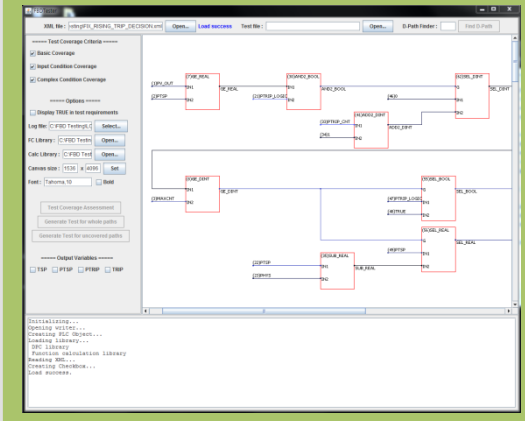
```

Verilog 2
C:\Users\Jonghoon\Desktop\FBD@the_cb_X_Prefig_1.v4.v
endfunction
endmodule
module TOP(clk, IN, DELAY, OUT);
input clk;
input IN;
input [0:6] DELAY;
output OUT;
reg[0:2] timer;
initial timer = 3'b000;
assign OUT = !IN && (timer ==
DELAY) ? 0;
always @(posedge clk) begin
if(!IN) begin
timer = 3'b001;
else
timer = DELAY;
end
else
timer = 3'b000;
end

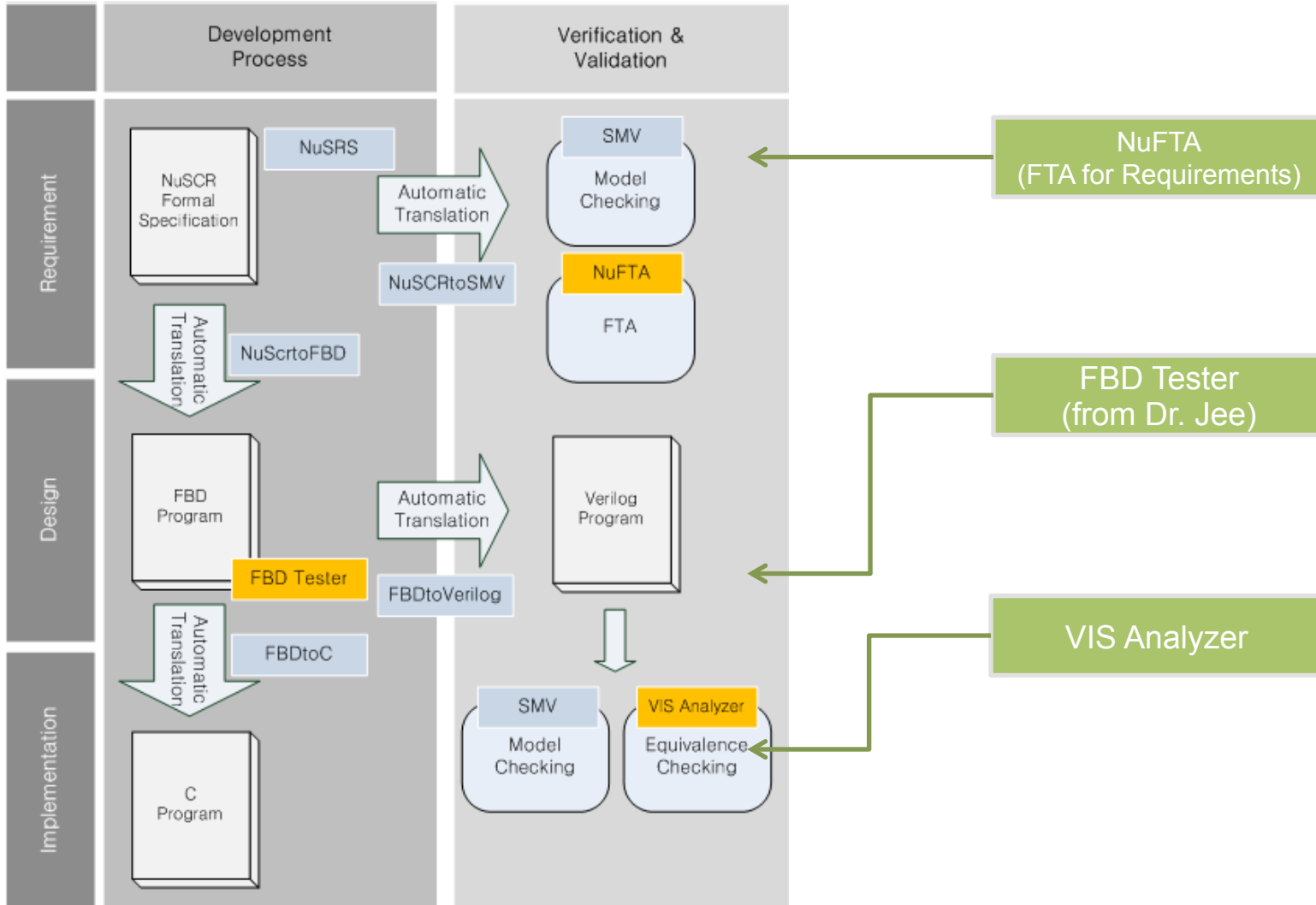
```

FBD Tester

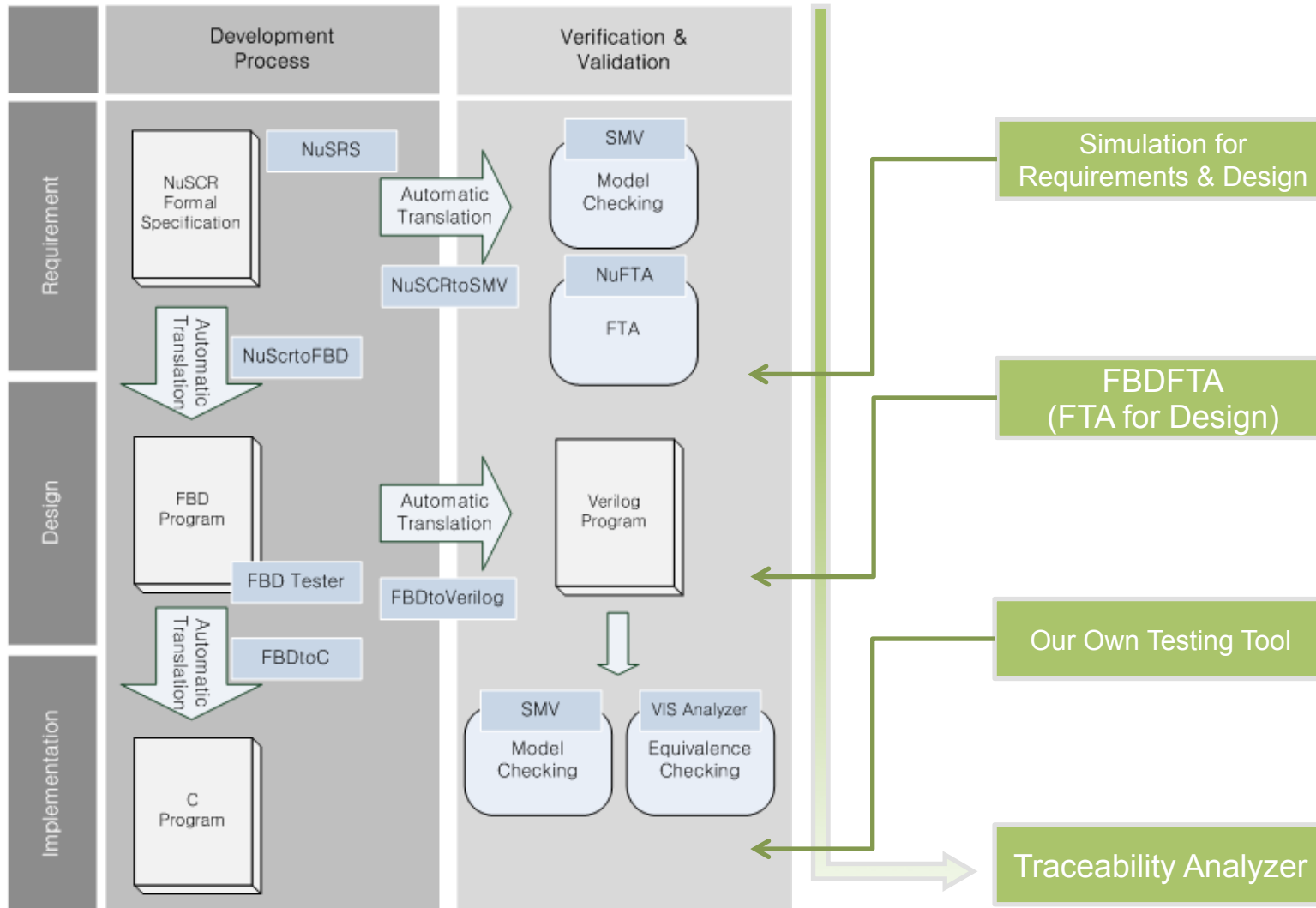
- Generate Test Cases for FBDs Automatically



Not Yet Integrated (Cont'd)



Not Yet Developed



NuDE: Development Environment for
Safety-Critical Software of Nuclear Power Plant

Future NuDE

Consideration for Future NuDE

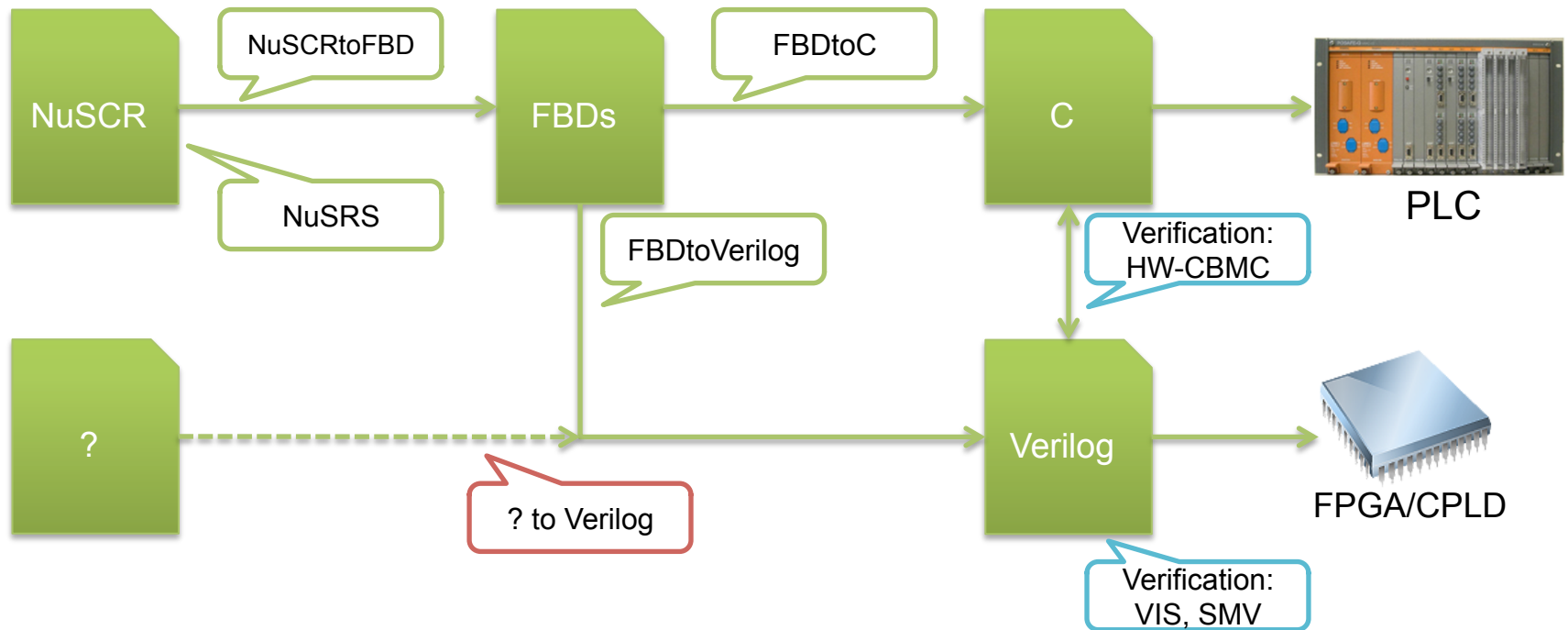
FBD Programming

- A Guide for Safe FBD Programming
 - How to Design FBD Program Safe?

IDE for NPP Software based on FPGA/CPLD

- Seamless Transition from PLC to FPGA/CPLD
 - Automatic Translation from FBD to Verilog (FBDtoVerilog)
- Dependable Development
 - Dependability Demonstration for FBDtoC and FBDtoVerilog
- Verification for FPGA/CPLD
 - Verification Techniques (Simulation, Testing, etc.)
- A All-New Formal Requirements Specification Method
 - Formal Requirements Specification for Verilog HDL

Future NuDE



NuDE: Development Environment for
Safety-Critical Software of Nuclear Power Plant

Conclusion

Conclusion

Our Goal

- SCADe를 능가할 수 있는 원자력 도메인 SW용 국산 IDE 개발
 - Dependable Development
 - Development life-cycle based on Formal Methods
 - Dependability Demonstration for Our Tools

Expectation

- 진화하는 원자력 SW 개발 환경을 선도
 - FPGA/CPLD기반의 SW 개발을 지원
 - PLC기반의 개발 산출물을 재사용
 - Natural Language Specification -> Formal Specification