KOREAN
NUCLEAR SOCIETY

www.kns.org

| 일 정 | 2014. 10. 29(수) ~ 10. 31(금)

| 장 소 | 보광 휘닉스파크

2014
추계학술발표회

A Correctness Verification Technique for Commercial FPGA Synthesis Tools

Eui-Sub Kim and Junbeom Yoo(KU)
Jong-Gyun Choi, Jang-Yeol Kim, and Jang-Soo Lee(KAERI)

Dependable Software Laboratory
Konkuk University

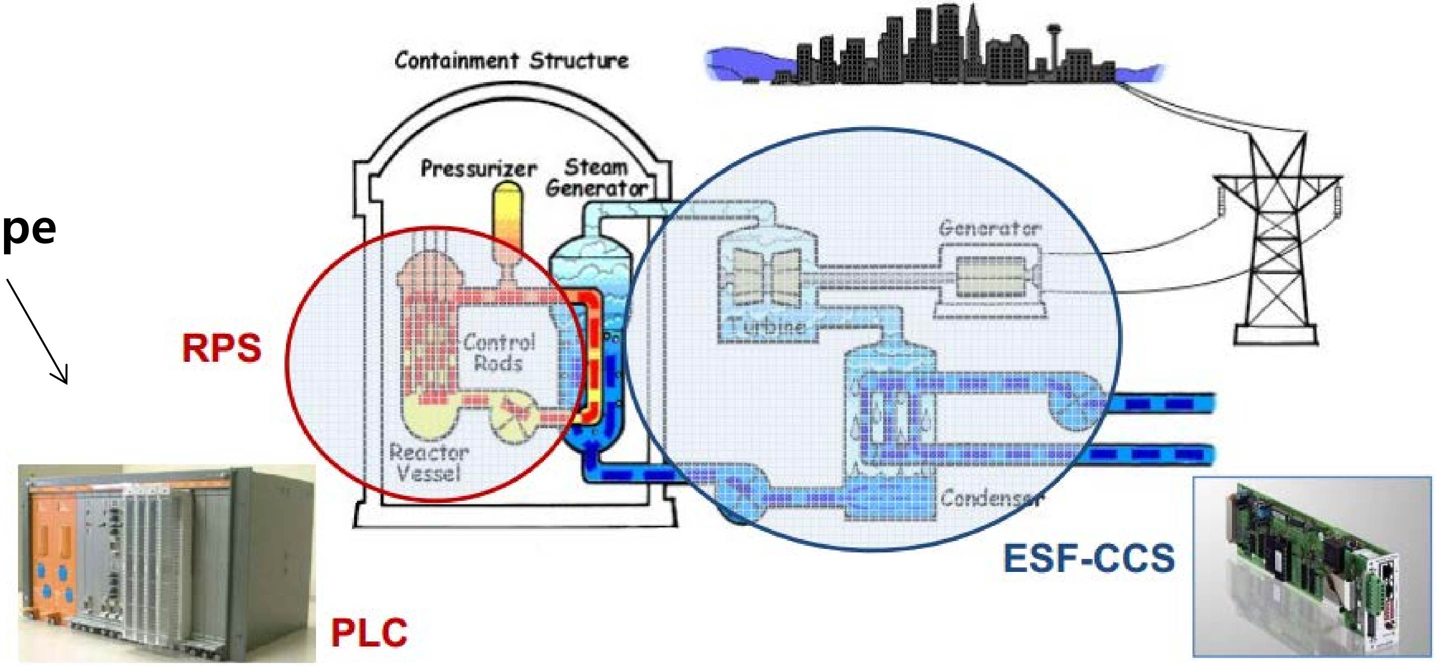
2014-11-03

Contents

- Introduction
- Background
- The Correctness Verification Technique
  - Indirect Verification approach
  - Formal Verification approach
  - EDIFtoBLIF-MV Translator
    - Constraints for VIS
    - Translation Rules of EDIFtoBLIF-MV
- Case Study
- Conclusion and Future work

Introduction

- Safety-Critical Software in Nuclear Power Plants
  - Reactor Protection System ➔ PLC (Programmable Logic Controller)



Scope

RPS

PLC

ESF-CCS

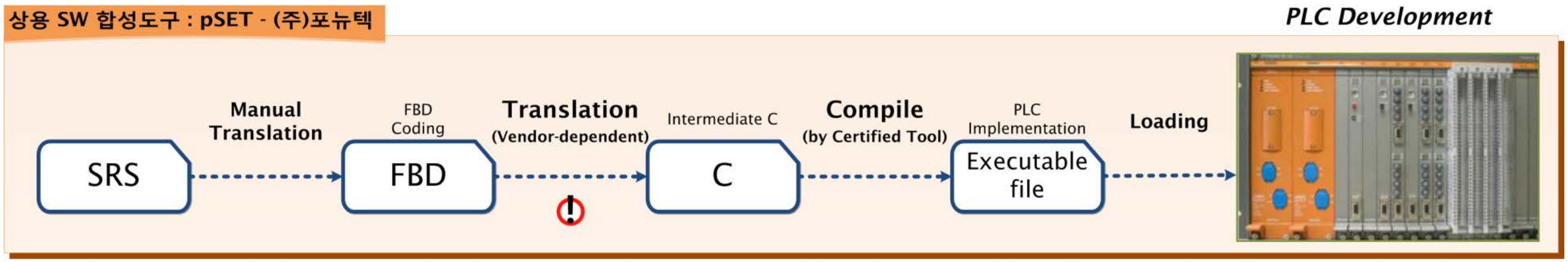
DCS

4

Introduction

❗ : " 안전성 증명 " 이 필요한 자동합성 부분

- Software Development Process based on PLC



상용 SW 합성도구 : pSET - (주)포뉴텍

PLC Development

SRS

Manual Translation

FBD Coding

FBD

Translation (Vendor-dependent)

❗

Intermediate C

C

Compile (by Certified Tool)

PLC Implementation

Executable file

Loading

PLC

FPGA

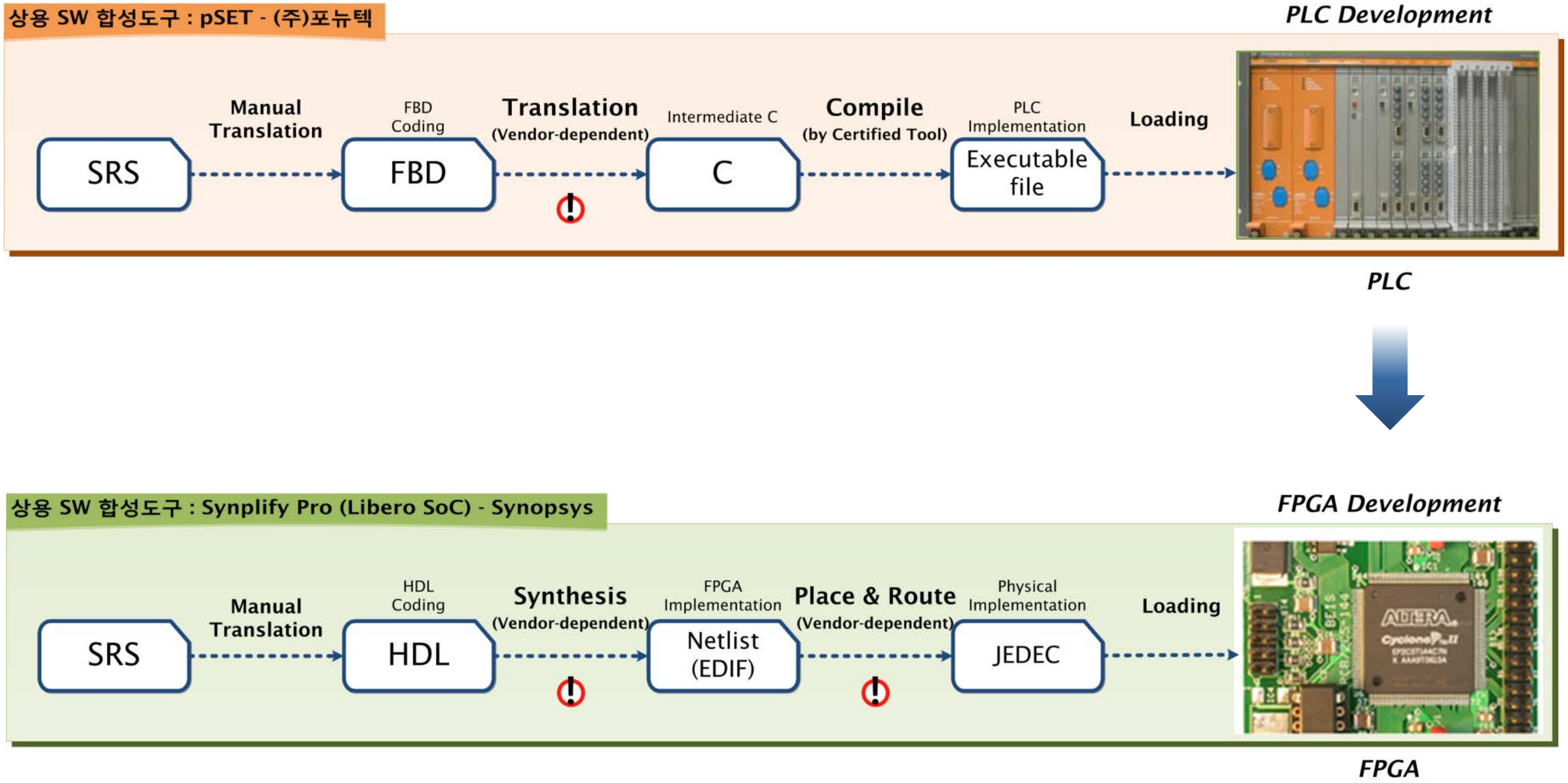
Recently, there are trend
to replace the platform from PLC to FPGA

Way????

Introduction

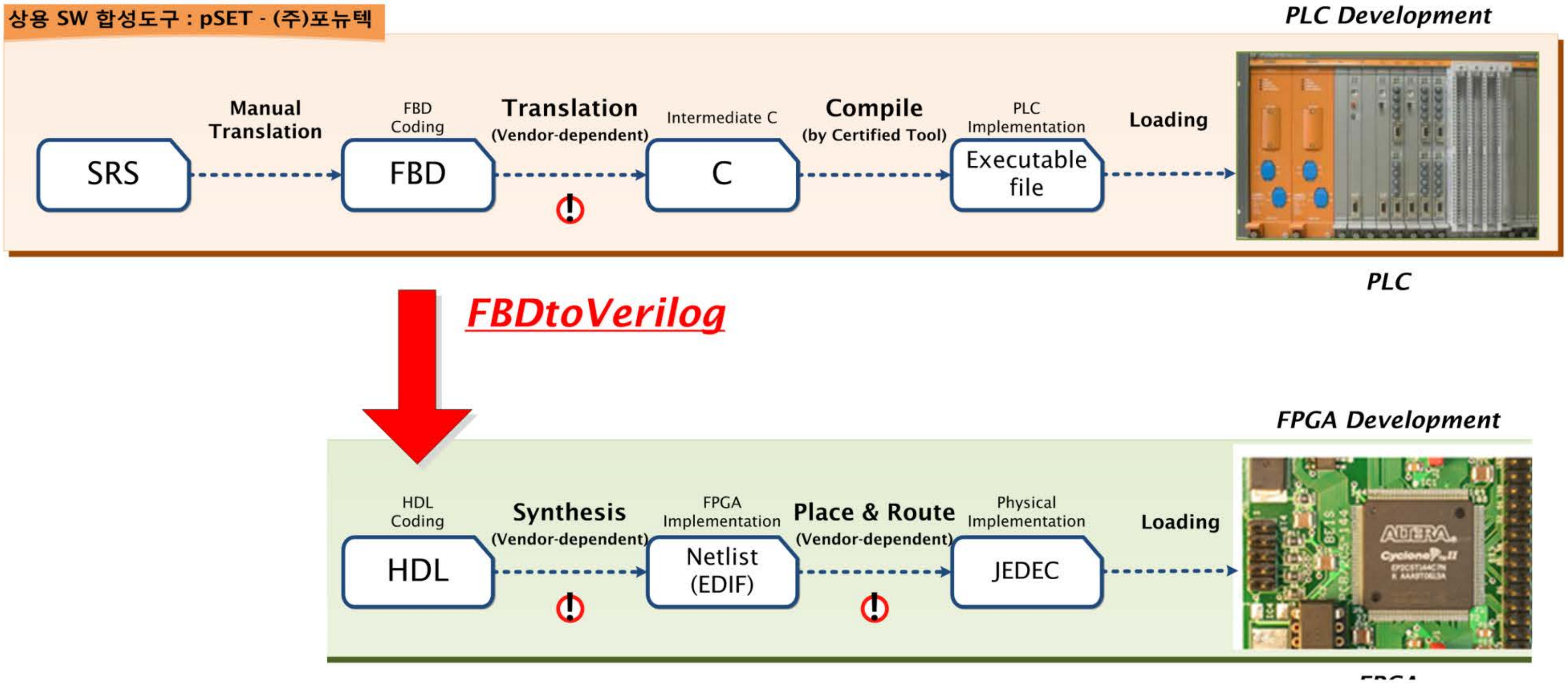
ⓘ : " 안전성 증명 " 이 필요한 자동합성 부분

- **PLC vs. FPGA**
  - There have differences in stage of software development process



Introduction

- We developed the **FBDtoVerilog** translator
  - It **automatically** translates an FBD to a Verilog program



상용 SW 합성도구 : pSET - (주)포뉴텍

PLC Development

SRS → Manual Translation → FBD Coding: FBD → Translation (Vendor-dependent) → Intermediate C: C → Compile (by Certified Tool) → PLC Implementation: Executable file → Loading → PLC

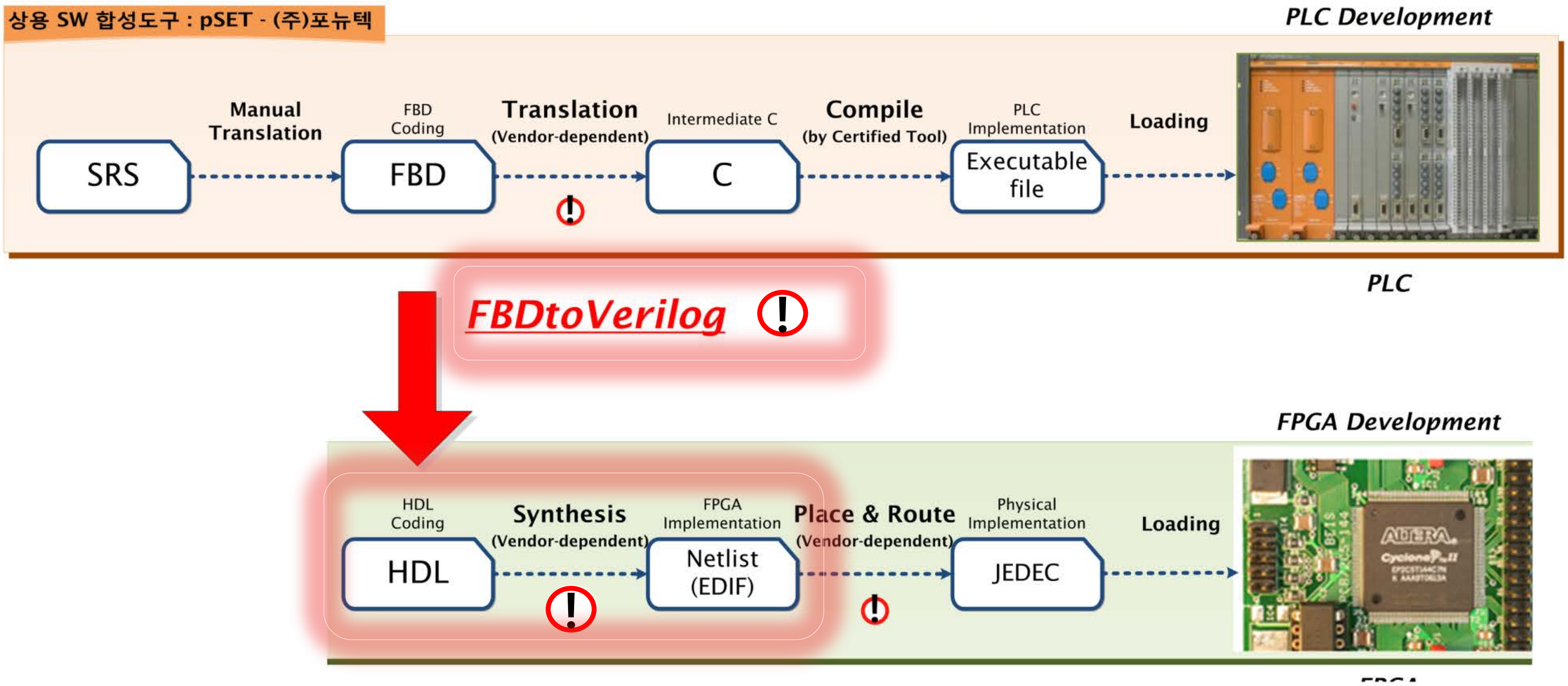
FBDtoVerilog

FPGA Development

HDL Coding: HDL → Synthesis (Vendor-dependent) → FPGA Implementation: Netlist (EDIF) → Place & Route (Vendor-dependent) → Physical Implementation: JEDEC → Loading → FPGA

Introduction

- We developed the **FBDtoVerilog** translator
  - It **automatically** translates an FBD to a Verilog program



상용 SW 합성도구 : pSET - (주)포뉴텍

PLC Development

SRS

Manual Translation

FBD Coding

FBD

Translation (Vendor-dependent) (!)

Intermediate C

C

Compile (by Certified Tool)

PLC Implementation

Executable file

Loading

PLC

FBDtoVerilog (!)

FPGA Development

HDL Coding

HDL

Synthesis (Vendor-dependent) (!)

FPGA Implementation

Netlist (EDIF)

Place & Route (Vendor-dependent) (!)

Physical Implementation

JEDEC

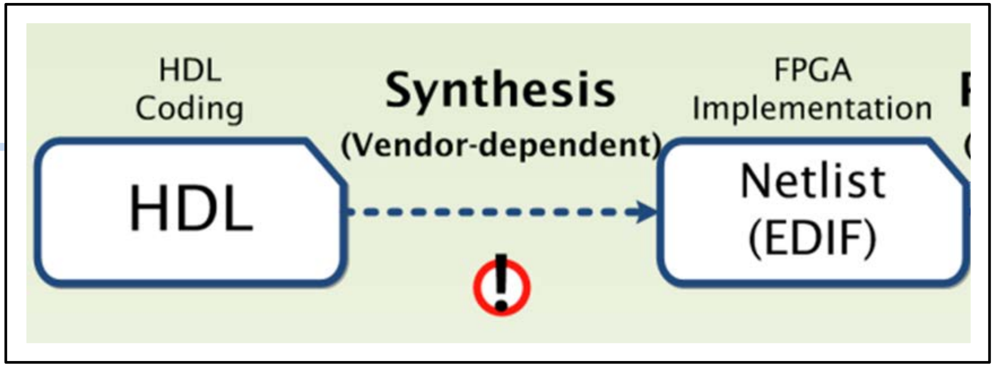
Loading

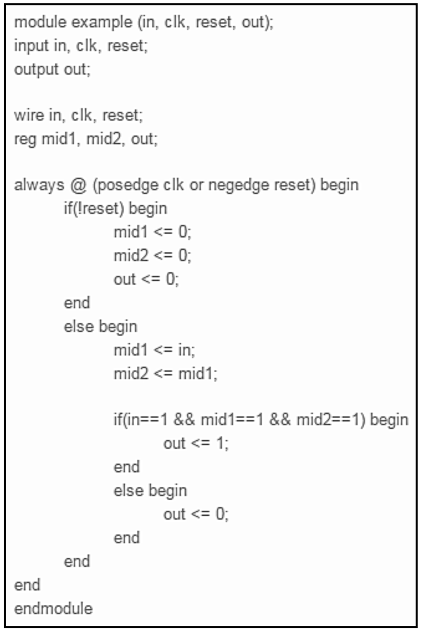
FPGA

(!) : " 안전성 증명 " 이 필요한 자동합성 부분

Background

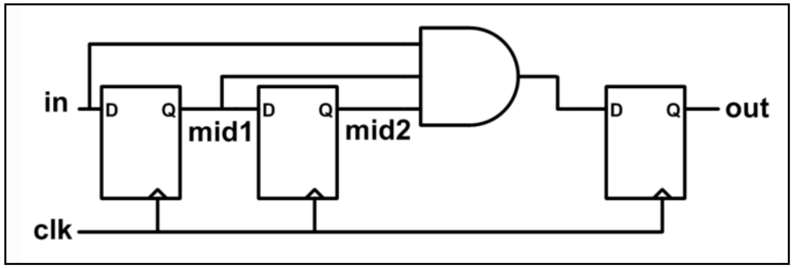
- Logic Synthesis
  - Register-transfer level
    ➔ Gate level





```
module example (in, clk, reset, out);
input in, clk, reset;
output out;

wire in, clk, reset;
reg mid1, mid2, out;

always @ (posedge clk or negedge reset) begin
    if(!reset) begin
        mid1 <= 0;
        mid2 <= 0;
        out <= 0;
    end
    else begin
        mid1 <= in;
        mid2 <= mid1;

        if(in==1 && mid1==1 && mid2==1) begin
            out <= 1;
        end
        else begin
            out <= 0;
        end
    end
end
endmodule
```

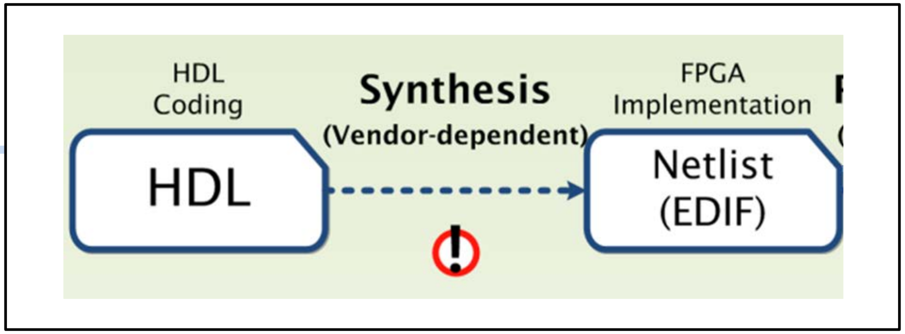
Synthesis



RTL (Register-transfer level)

Gate level

Background



- Commercial FPGA Synthesis Tools
  - 현재 다양한 3-rd parties 에 의해 개발된 Synthesis Tools 가 존재
  - Synthesis 는 복잡한 과정이 포함되어 있음
    - **Synthesis : circuit 의 area, power, performance 등을 높이기 위해 다양한 전략 및 최적화가 수행됨**
  - 기존 상용 Synthesis tool 들이 일반적으로는 좋은 성능을 보여 주지만, 신뢰성 ?, Certification 등의 문제 존재
    - **→ 따라서 철저하고 엄격한 방법으로 correctness 를 Demonstration / verification 할 필요가 있음**
  - Vendors

SYNOPSYS®

cadence

XILINX ISE

XILINX®

Mentor Graphics®

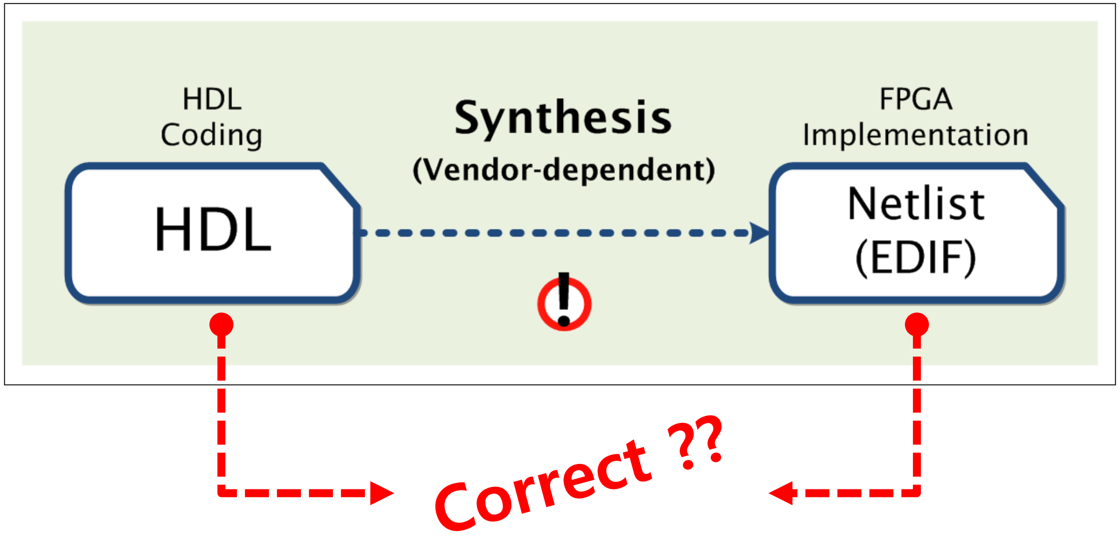
ALTERA QUARTUS II

MAGMA

LATTICE SEMICONDUCTOR

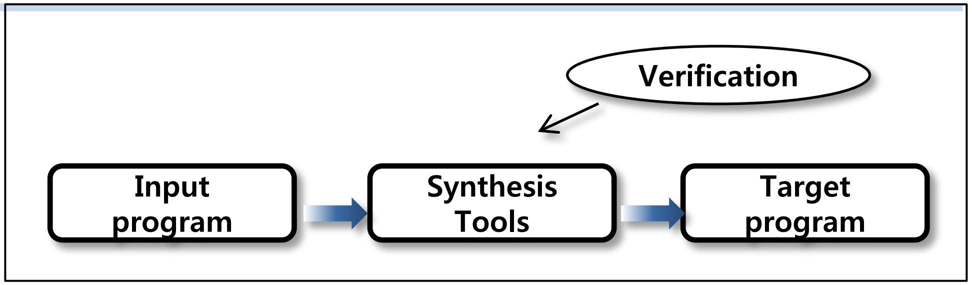
The Correctness Verification Technique

- Proposed Correctness Verification Technique
  - **1) Indirect verification approach**
  - **2) Formal verification technique**



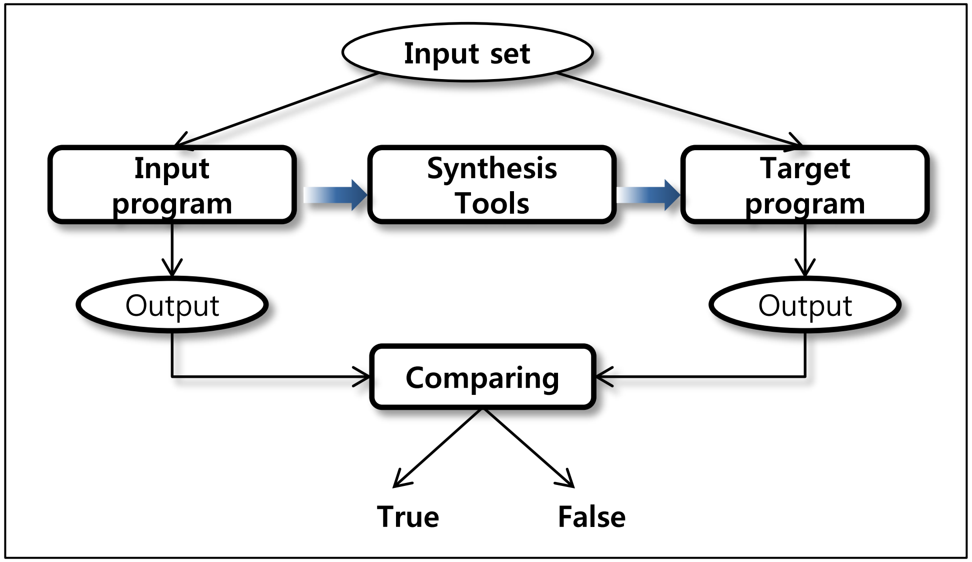
Indirect verification

- Direct Verification
  - 변환기 자체의 검증



Direct Verification

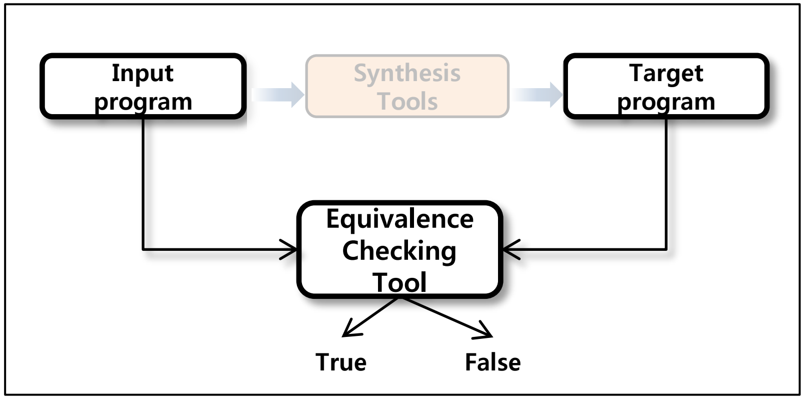
- **Indirect Verification**
  - 변환 전 program 과 변환 후 프로그램이
  - 동일한 기능을 하는지 검증
  - **적어도 주어진 Logic과 변환된 Logic 이 일치한다는 것을 증명**



Indirect Verification

Formal Verification

- **Equivalence Checking**
  - this proves that two given design have the same functionality



Input program

Synthesis Tools

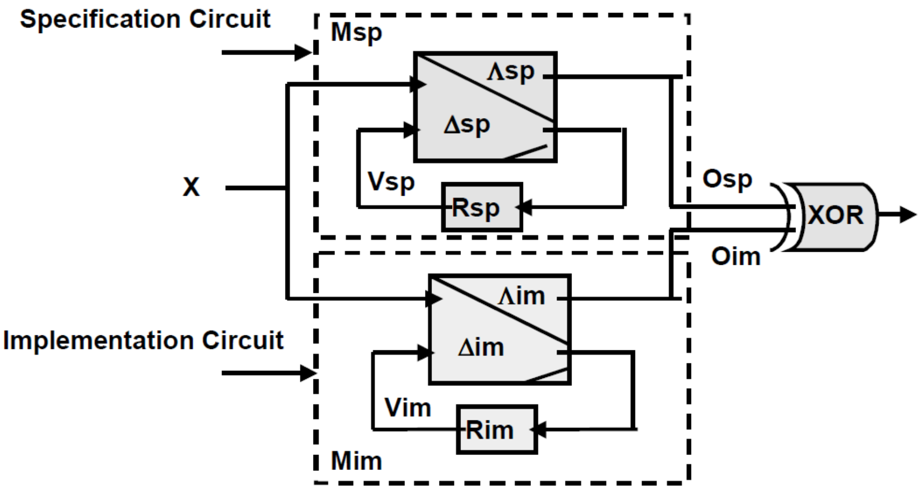
Target program

Equivalence Checking Tool

True

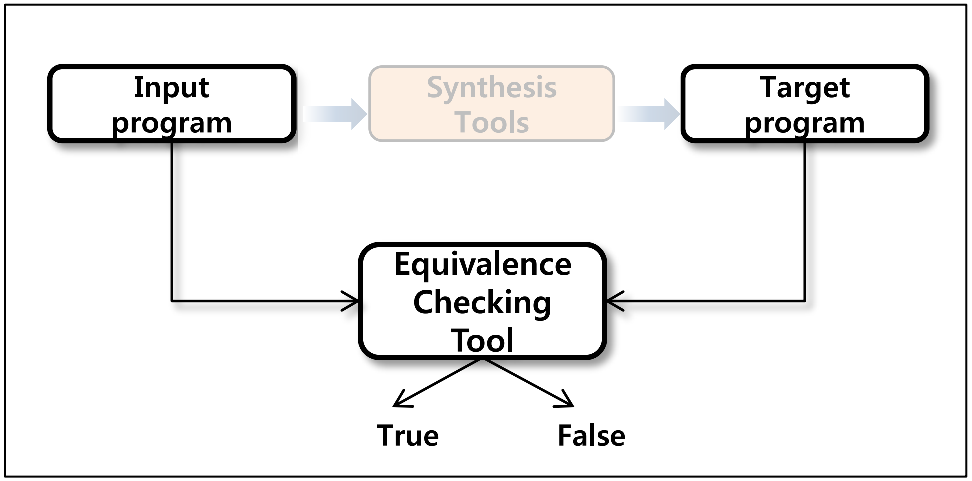
False

Equivalence Checking



Formal Verification

- Commercial Equivalence Checking Tools



Equivalence Checking

VIS

cādence

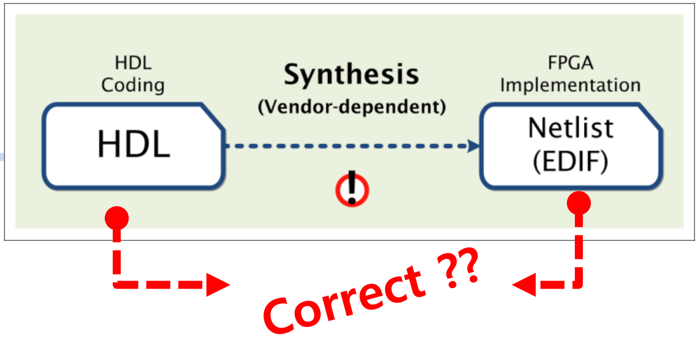
Mentor Graphics

PS

SYNOPSYS®

MAGMA

VIS



HDL Coding

HDL

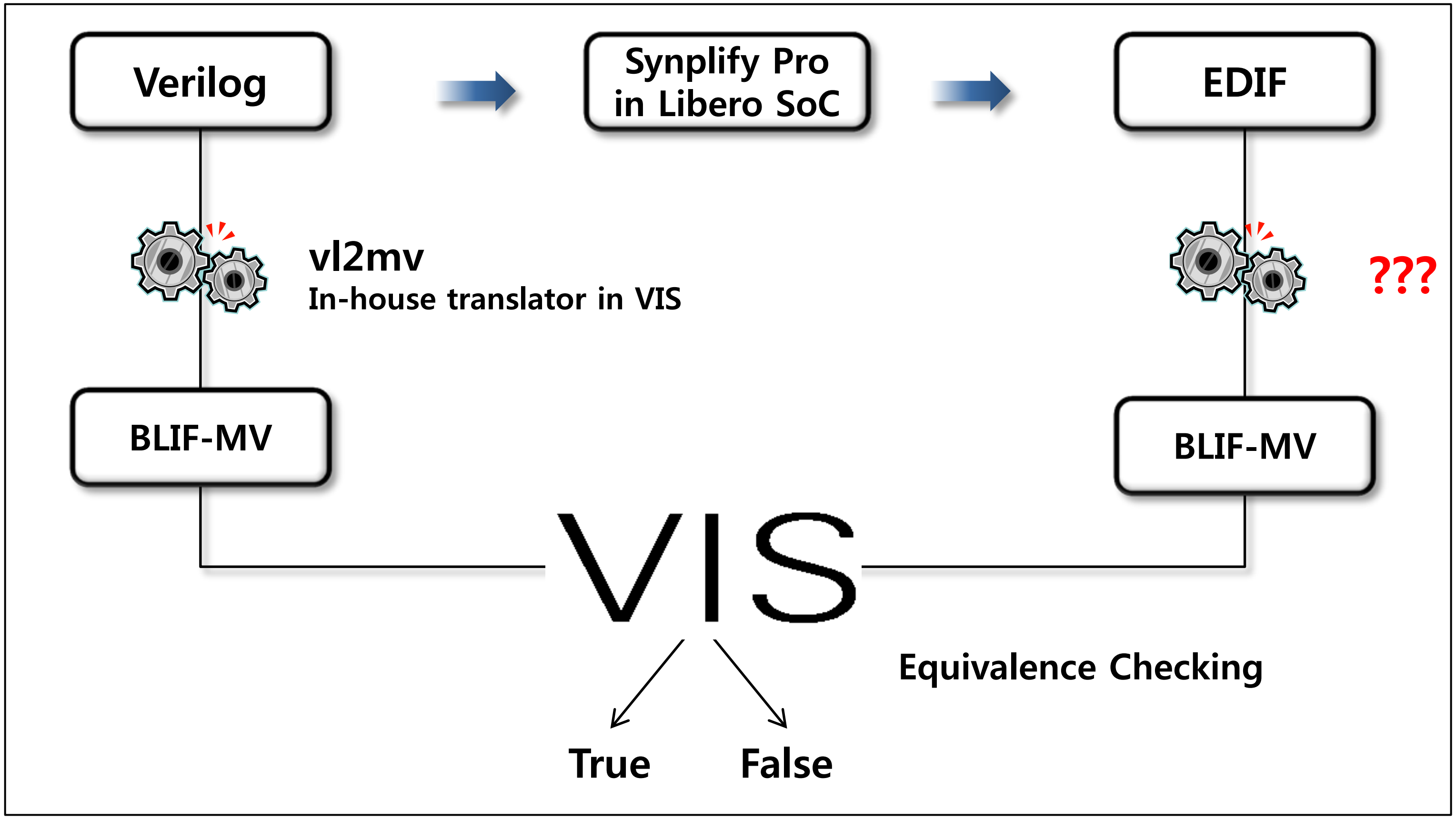
Synthesis (Vendor-dependent)

FPGA Implementation

Netlist (EDIF)

Correct ??

- VIS의 front-end language ➔ BLIF-MV



Verilog

Synplify Pro in Libero SoC

EDIF

vl2mv

In-house translator in VIS

???

BLIF-MV

BLIF-MV

VIS

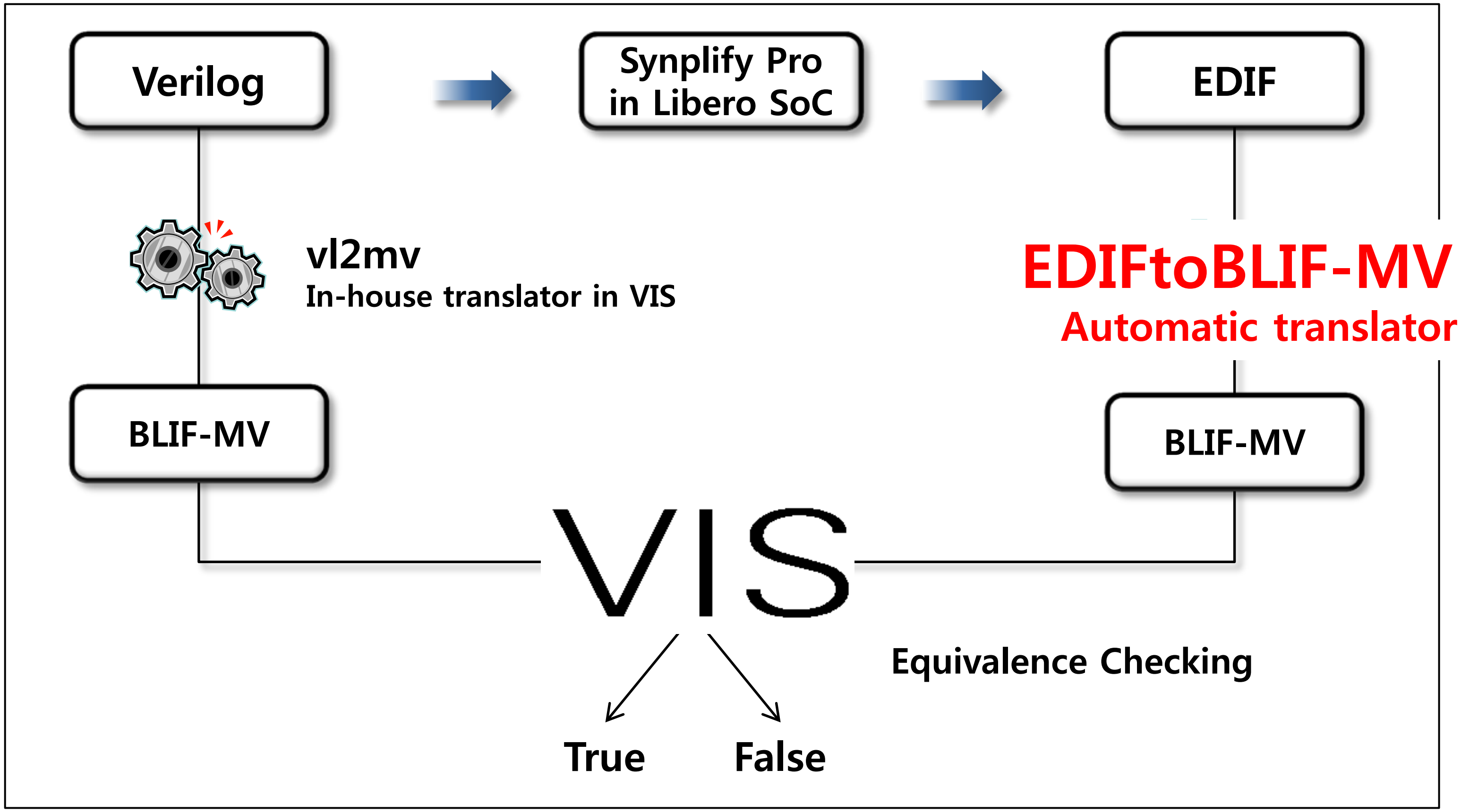
Equivalence Checking

True

False

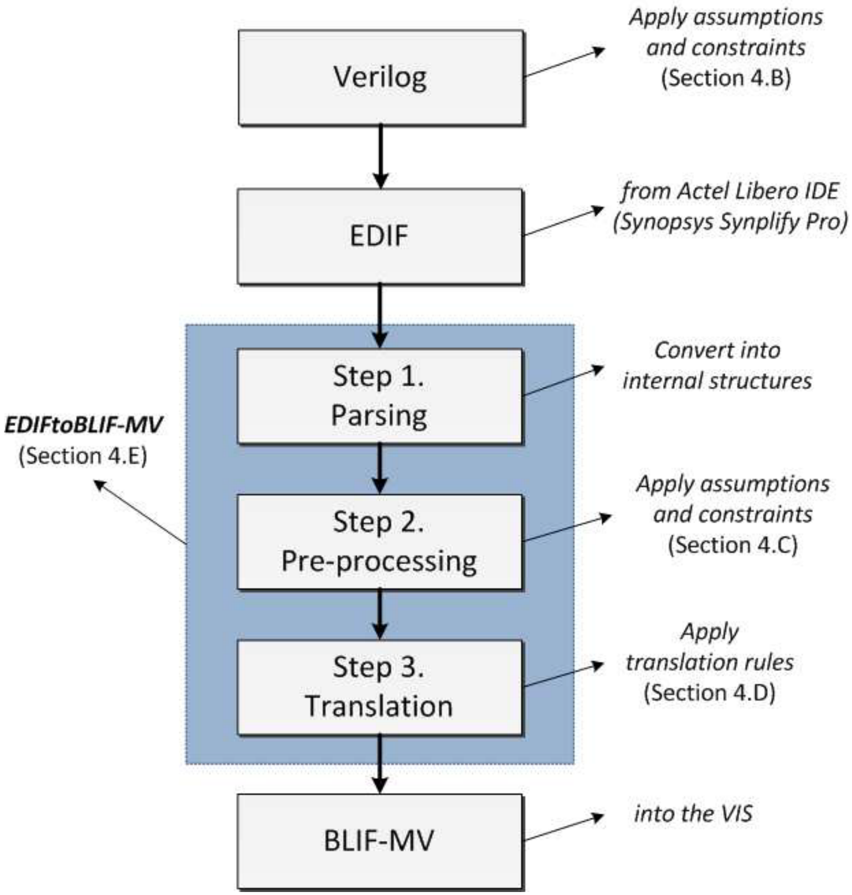
EDIFtoBLIF – MV

- EDIF 를 BLIF-MV 로 변환해 주는 EDIFtoBLIF-MV 변환기 개발



Process of EDIFtoBLIF – MV

- The Model Transformation from EDIF to BLIF-MV



Verilog

Apply assumptions and constraints (Section 4.B)

EDIF

from Actel Libero IDE (Synopsys Synplify Pro)

Step 1. Parsing

Convert into internal structures

EDIFtoBLIF-MV (Section 4.E)

Step 2. Pre-processing

Apply assumptions and constraints (Section 4.C)

Step 3. Translation

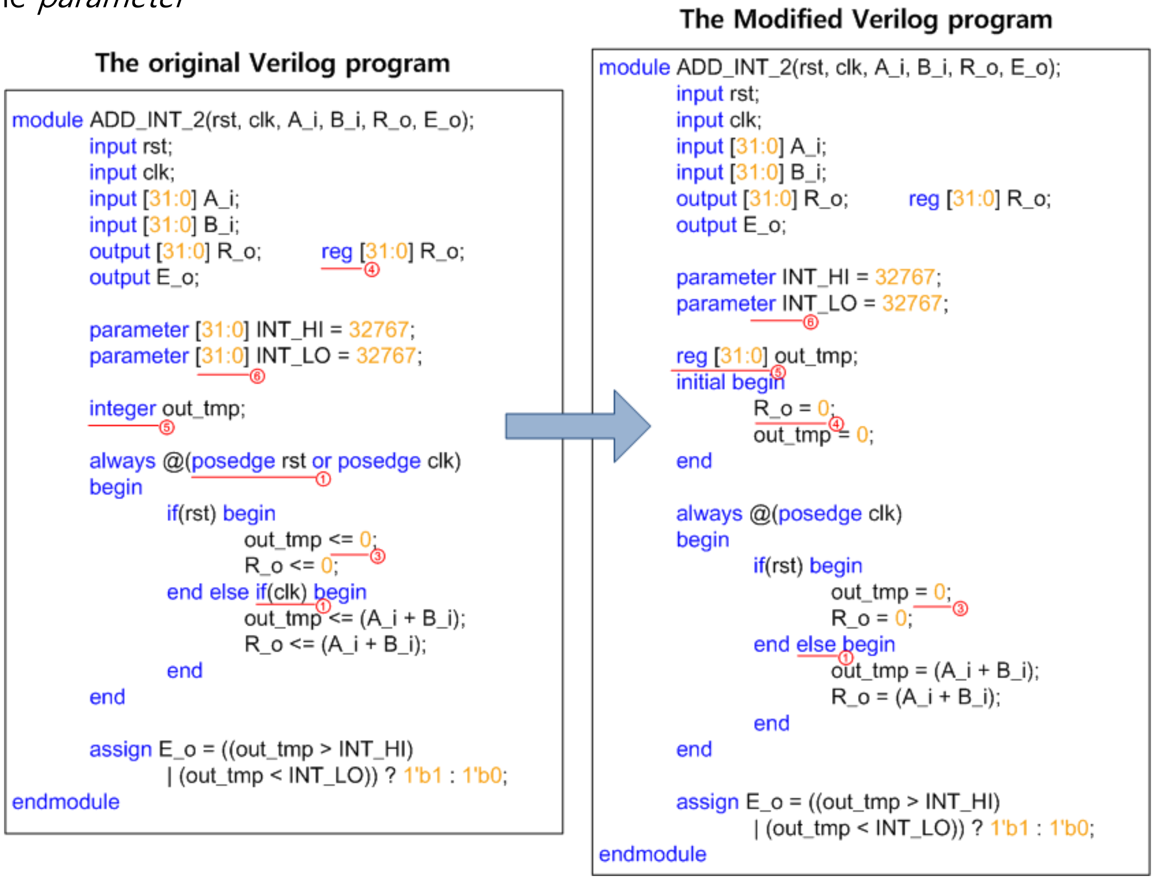
Apply translation rules (Section 4.D)

BLIF-MV

into the VIS

Vis constraints

- Vis constraints
  - 1) Use the clock *clk* only at the statement *always*
  - *@(posedge clk)*
  - 2) Do not use the time delay
  - 3) Do not use the non-blocking statement
  - 4) All *reg* variables should be initialized with 0
  - 5) Do not use the *integer* typed variable
  - 6) Do not use the size of bits to define *parameter*



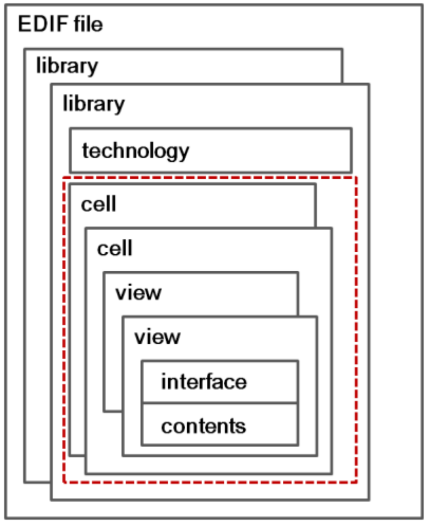
The original Verilog program

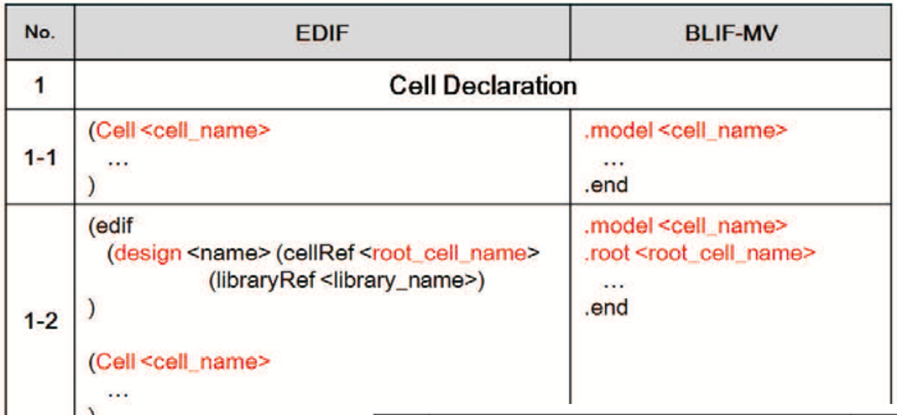
```
module ADD_INT_2(rst, clk, A_i, B_i, R_o, E_o);
        input rst;
        input clk;
        input [31:0] A_i;
        input [31:0] B_i;
        output [31:0] R_o;        reg [31:0] R_o;   ④
        output E_o;

        parameter [31:0] INT_HI = 32767;
        parameter [31:0] INT_LO = 32767;   ⑥

        integer out_tmp;   ⑤

        always @(posedge rst or posedge clk)   ①
        begin
                if(rst) begin
                        out_tmp <= 0;
                        R_o <= 0;   ③
                end else if(clk) begin   ①
                        out_tmp <= (A_i + B_i);
                        R_o <= (A_i + B_i);
                end
        end

        assign E_o = ((out_tmp > INT_HI)
                | (out_tmp < INT_LO)) ? 1'b1 : 1'b0;
endmodule
```

The Modified Verilog program

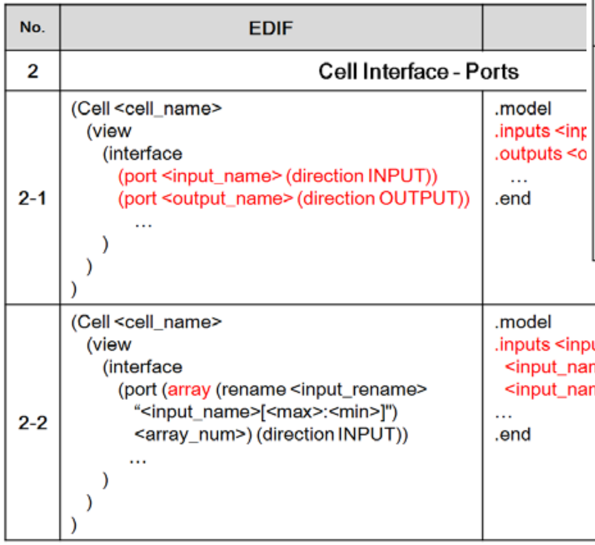
```
module ADD_INT_2(rst, clk, A_i, B_i, R_o, E_o);
        input rst;
        input clk;
        input [31:0] A_i;
        input [31:0] B_i;
        output [31:0] R_o;        reg [31:0] R_o;
        output E_o;

        parameter INT_HI = 32767;
        parameter INT_LO = 32767;   ⑥

        reg [31:0] out_tmp;   ⑤
        initial begin
                R_o = 0;   ④
                out_tmp = 0;
        end

        always @(posedge clk)
        begin
                if(rst) begin
                        out_tmp = 0;
                        R_o = 0;   ③
                end else begin   ①
                        out_tmp = (A_i + B_i);
                        R_o = (A_i + B_i);
                end
        end

        assign E_o = ((out_tmp > INT_HI)
                | (out_tmp < INT_LO)) ? 1'b1 : 1'b0;
endmodule
```

Translation Rule of EDIFtoBLIF – MV

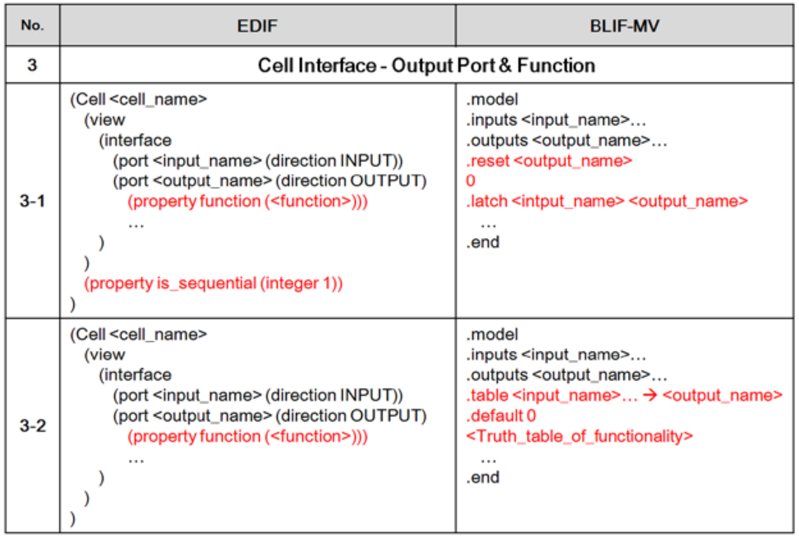




No.	EDIF	BLIF-MV
1	Cell Declaration	
1-1	(Cell <cell_name> …)	.model <cell_name> … .end
1-2	(edif (design <name> (cellRef <root_cell_name> (libraryRef <library_name>)) (Cell <cell_name> …)	.model <cell_name> .root <root_cell_name> … .end

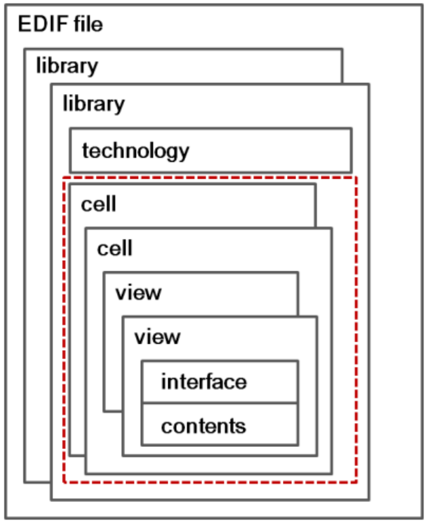


No.	EDIF	
2	Cell Interface - Ports	
2-1	(Cell <cell_name> (view (interface (port <input_name> (direction INPUT)) (port <output_name> (direction OUTPUT)) …)))	.model .inputs <inp .outputs <o … .end
2-2	(Cell <cell_name> (view (interface (port (array (rename <input_rename> "<input_name>[<max>:<min>]") <array_num>) (direction INPUT)) …)))	.model .inputs <input_name><<min>> <input_name><<min+1>> … <input_name><<max>> … .end



No.	EDIF	BLIF-MV
3	Cell Interface - Output Port & Function	
3-1	(Cell <cell_name> (view (interface (port <input_name> (direction INPUT)) (port <output_name> (direction OUTPUT) (property function (<function>))) …)) (property is_sequential (integer 1)))	.model .inputs <input_name>… .outputs <output_name>… .reset <output_name> 0 .latch <intput_name> <output_name> … .end
3-2	(Cell <cell_name> (view (interface (port <input_name> (direction INPUT)) (port <output_name> (direction OUTPUT) (property function (<function>))) …)))	.model .inputs <input_name>… .outputs <output_name>… .table <input_name>… → <output_name> .default 0 <Truth_table_of_functionality> … .end

Translation Rule of EDIFtoBLIF－MV



No.	EDIF	BLIF-MV
4	Truth Table of Functionality with Examples	
4-1	property function (string "1")	.table → output =1
4-2	property function (string "A + B")	.table A B → output .default 0 0 0 0 1 0 1 0 1 1 1 1 1
4-3	property function (string "A + B +C")	.table A B C → output .default 0 0 0 0 0 0 0 1 1 0 1 0 1 0 1 1 1 1 0 0 1 1 0 1 1 1 1 0 1 1 1 1 1

Description

Content - instance

(viewRef <viewRef_name>
> (libraryRef <libraryRef_name>)))

)
)

BLIF-MV

```
.model
  …
.mv <instance_name>, <instance_name + _intput_name_of_cellRef>, …

.subckt <cellRef_name> <instance_name + "_subckt">
      <intput_name_of_cellRef> = <instance_name + _intput_name_of_cellRef>
        …
     <output_name_of_cellRef> = <instance_name>
  …
.end
```

No.

De

6

Cell C

EDIF

```
(Cell <cell_name>
  (view
    (contents
      (net <net_name> (joined
        (portRef <input_port> (instanceRef <input_instance_name>))
        (portRef <output_port> (instanceRef <output_instance_name>))
          …
       (portRef <output_port> (instanceRef <output_instance_name>))
```

name><input_port> → <output_instance_name>_<output_port>
ame>

name><input_port> → <output_instance_name>_<output_port>
ame>

re_net_name> "<original_name>[order]") (joined

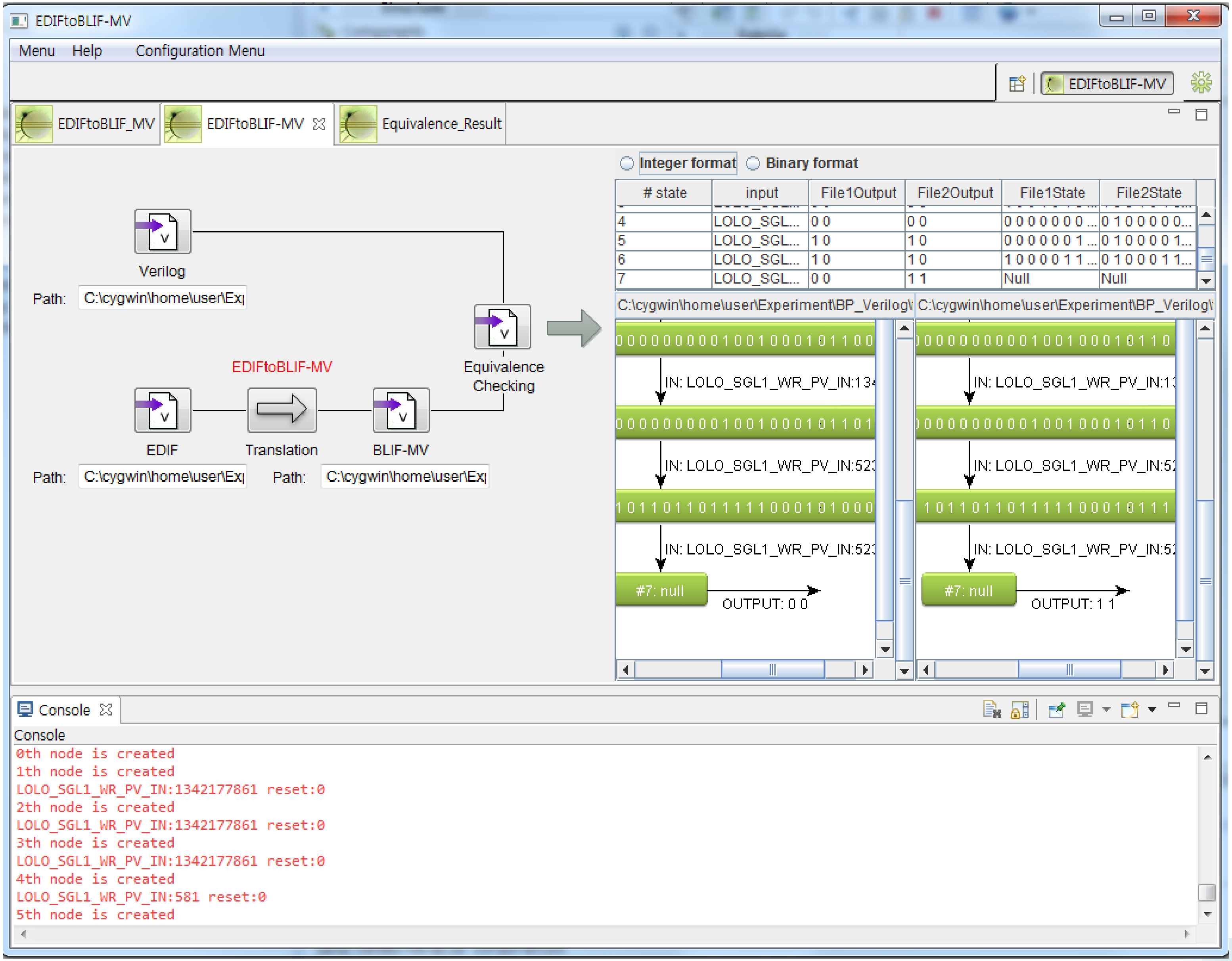
mber <member_name> <order>)

)

BLIF -MV

```
.model
  …
.table … <original_name>_<order> …

.end
```

EDIFtoBLIF-MV Translator + Equivalence Checking

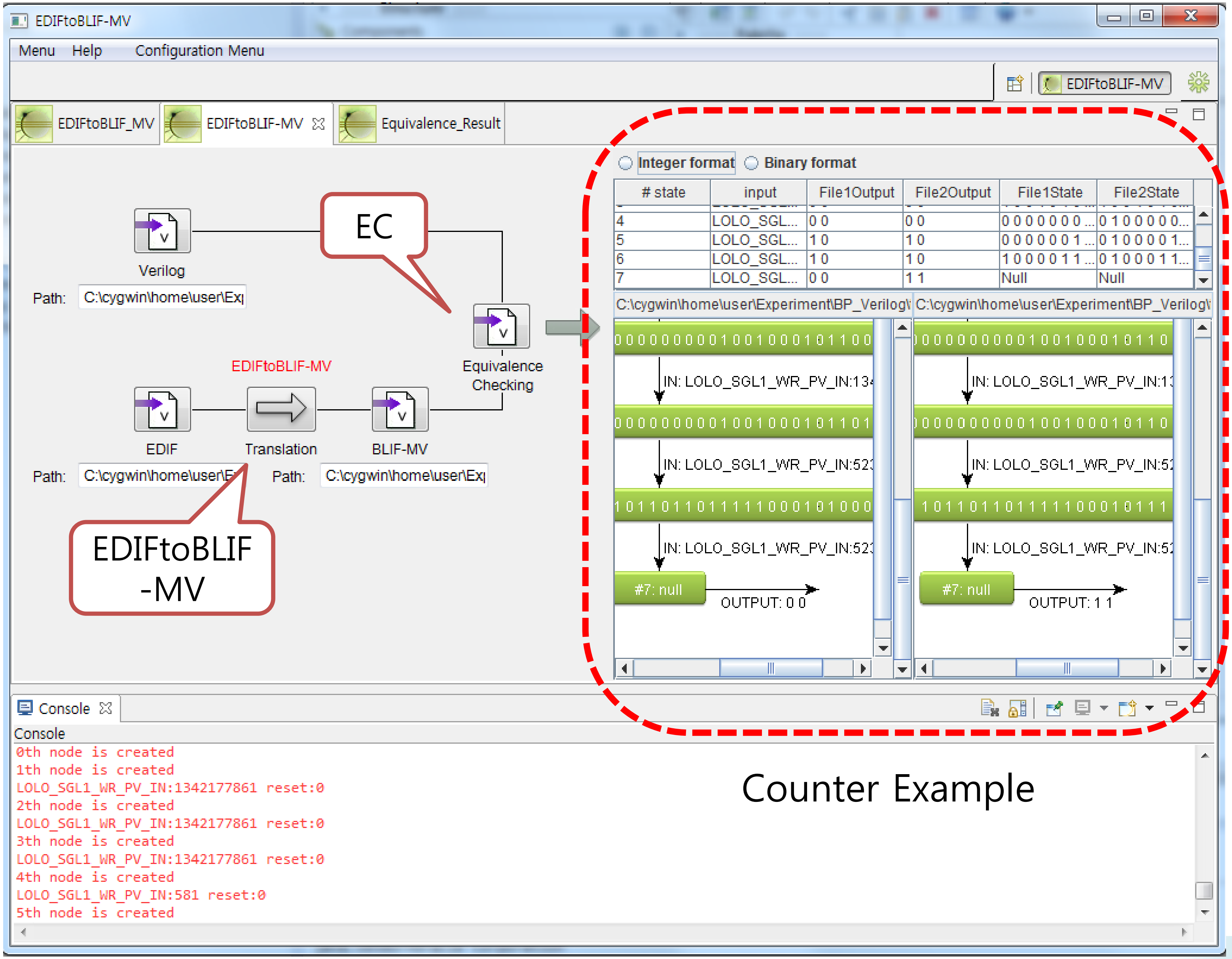


# state	input	File1Output	File2Output	File1State	File2State
4	LOLO_SGL...	0 0	0 0	0 0 0 0 0 0 0 ...	0 1 0 0 0 0 0...
5	LOLO_SGL...	1 0	1 0	0 0 0 0 0 0 1 ...	0 1 0 0 0 0 1...
6	LOLO_SGL...	1 0	1 0	1 0 0 0 0 1 1 ...	0 1 0 0 0 1 1...
7	LOLO_SGL...	0 0	1 1	Null	Null

Console

```
0th node is created
1th node is created
LOLO_SGL1_WR_PV_IN:1342177861 reset:0
2th node is created
LOLO_SGL1_WR_PV_IN:1342177861 reset:0
3th node is created
LOLO_SGL1_WR_PV_IN:1342177861 reset:0
4th node is created
LOLO_SGL1_WR_PV_IN:581 reset:0
5th node is created
```

EDIFtoBLIF－MV Translator + Equivalence Checking tool



Path: C:\cygwin\home\user\Ex|

Path: C:\cygwin\home\user\Ex|

Path: C:\cygwin\home\user\Ex|

○ Integer format ○ Binary format

# state	input	File1Output	File2Output	File1State	File2State
4	LOLO_SGL...	0 0	0 0	0 0 0 0 0 0 0 ...	0 1 0 0 0 0 0...
5	LOLO_SGL...	1 0	1 0	0 0 0 0 0 0 1 ...	0 1 0 0 0 0 1...
6	LOLO_SGL...	1 0	1 0	1 0 0 0 0 1 1 ...	0 1 0 0 0 1 1...
7	LOLO_SGL...	0 0	1 1	Null	Null

C:\cygwin\home\user\Experiment\BP_Verilog\

0 0 0 0 0 0 0 0 0 1 0 0 1 0 0 0 1 0 1 1 0 0

0 0 0 0 0 0 0 0 0 1 0 0 1 0 0 0 1 0 1 1 0 1

1 0 1 1 0 1 1 0 1 1 1 1 1 0 0 0 1 0 1 0 0 0

C:\cygwin\home\user\Experiment\BP_Verilog\

0 0 0 0 0 0 0 0 0 0 1 0 0 1 0 0 0 1 0 1 1 0

0 0 0 0 0 0 0 0 0 0 1 0 0 1 0 0 0 1 0 1 1 0

1 0 1 1 0 1 1 0 1 1 1 1 1 0 0 0 1 0 1 1 1

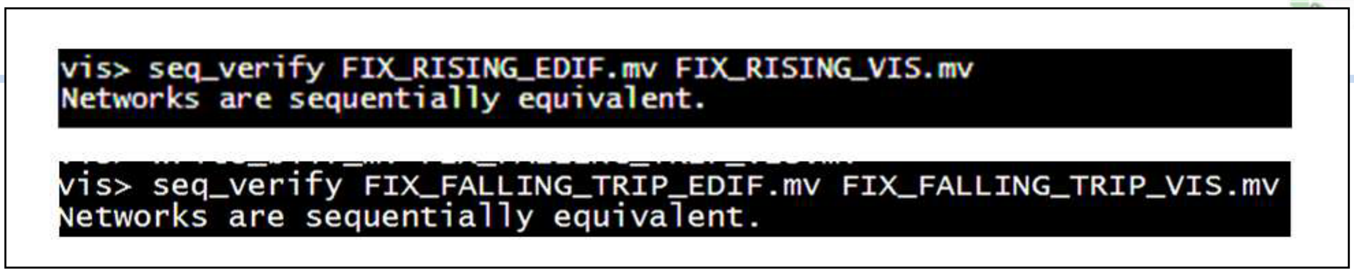
Console

```
0th node is created
1th node is created
LOLO_SGL1_WR_PV_IN:1342177861 reset:0
2th node is created
LOLO_SGL1_WR_PV_IN:1342177861 reset:0
3th node is created
LOLO_SGL1_WR_PV_IN:1342177861 reset:0
4th node is created
LOLO_SGL1_WR_PV_IN:581 reset:0
5th node is created
```

EC

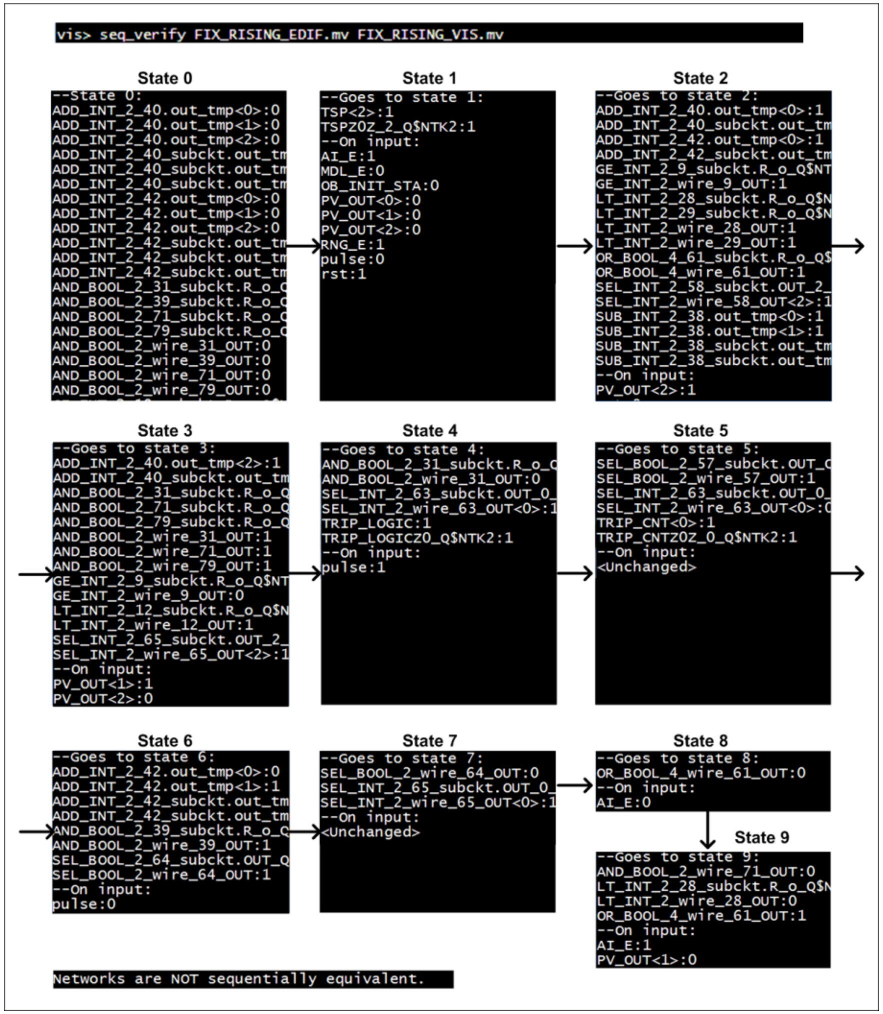
EDIFtoBLIF -MV

Counter Example



```
vis> seq_verify FIX_RISING_EDIF.mv FIX_RISING_VIS.mv
Networks are sequentially equivalent.
```

```
vis> seq_verify FIX_FALLING_TRIP_EDIF.mv FIX_FALLING_TRIP_VIS.mv
Networks are sequentially equivalent.
```



```
vis> seq_verify FIX_RISING_EDIF.mv FIX_RISING_VIS.mv
```

State 0

```
--State 0:
ADD_INT_2_40.out_tmp<0>:0
ADD_INT_2_40.out_tmp<1>:0
ADD_INT_2_40.out_tmp<2>:0
ADD_INT_2_40_subckt.out_tm
ADD_INT_2_40_subckt.out_tm
ADD_INT_2_40_subckt.out_tm
ADD_INT_2_42.out_tmp<0>:0
ADD_INT_2_42.out_tmp<1>:0
ADD_INT_2_42.out_tmp<2>:0
ADD_INT_2_42_subckt.out_tm
ADD_INT_2_42_subckt.out_tm
ADD_INT_2_42_subckt.out_tm
AND_BOOL_2_31_subckt.R_o_Q
AND_BOOL_2_39_subckt.R_o_Q
AND_BOOL_2_71_subckt.R_o_Q
AND_BOOL_2_79_subckt.R_o_Q
AND_BOOL_2_wire_31_OUT:0
AND_BOOL_2_wire_39_OUT:0
AND_BOOL_2_wire_71_OUT:0
AND_BOOL_2_wire_79_OUT:0
```

State 1

```
--Goes to state 1:
TSP<2>:1
TSPZOZ_2_Q$NTK2:1
--On input:
AI_E:1
MDL_E:0
OB_INIT_STA:0
PV_OUT<0>:0
PV_OUT<1>:0
PV_OUT<2>:0
RNG_E:1
pulse:0
rst:1
```

State 2

```
--Goes to state 2:
ADD_INT_2_40.out_tmp<0>:1
ADD_INT_2_40_subckt.out_tm
ADD_INT_2_42.out_tmp<0>:1
ADD_INT_2_42_subckt.out_tm
GE_INT_2_9_subckt.R_o_Q$NT
GE_INT_2_wire_9_OUT:1
LT_INT_2_28_subckt.R_o_Q$N
LT_INT_2_29_subckt.R_o_Q$N
LT_INT_2_wire_28_OUT:1
LT_INT_2_wire_29_OUT:1
OR_BOOL_4_61_subckt.R_o_Q$
OR_BOOL_4_wire_61_OUT:1
SEL_INT_2_58_subckt.OUT_2_
SEL_INT_2_wire_58_OUT<2>:1
SUB_INT_2_38.out_tmp<0>:1
SUB_INT_2_38.out_tmp<1>:1
SUB_INT_2_38_subckt.out_tm
SUB_INT_2_38_subckt.out_tm
--On input:
PV_OUT<2>:1
```

State 3

```
--Goes to state 3:
ADD_INT_2_40.out_tmp<2>:1
ADD_INT_2_40_subckt.out_tm
AND_BOOL_2_31_subckt.R_o_Q
AND_BOOL_2_71_subckt.R_o_Q
AND_BOOL_2_79_subckt.R_o_Q
AND_BOOL_2_wire_31_OUT:1
AND_BOOL_2_wire_71_OUT:1
AND_BOOL_2_wire_79_OUT:1
GE_INT_2_9_subckt.R_o_Q$NT
GE_INT_2_wire_9_OUT:0
LT_INT_2_12_subckt.R_o_Q$N
LT_INT_2_wire_12_OUT:1
SEL_INT_2_65_subckt.OUT_2_
SEL_INT_2_wire_65_OUT<2>:1
--On input:
PV_OUT<1>:1
PV_OUT<2>:0
```

State 4

```
--Goes to state 4:
AND_BOOL_2_31_subckt.R_o_Q
AND_BOOL_2_wire_31_OUT:0
SEL_INT_2_63_subckt.OUT_0_
SEL_INT_2_wire_63_OUT<0>:1
TRIP_LOGIC:1
TRIP_LOGICZO_Q$NTK2:1
--On input:
pulse:1
```

State 5

```
--Goes to state 5:
SEL_BOOL_2_57_subckt.OUT_Q
SEL_BOOL_2_wire_57_OUT:1
SEL_INT_2_63_subckt.OUT_0_
SEL_INT_2_wire_63_OUT<0>:0
TRIP_CNT<0>:1
TRIP_CNTZOZ_0_Q$NTK2:1
--On input:
<Unchanged>
```

State 6

```
--Goes to state 6:
ADD_INT_2_42.out_tmp<0>:0
ADD_INT_2_42.out_tmp<1>:1
ADD_INT_2_42_subckt.out_tm
ADD_INT_2_42_subckt.out_tm
AND_BOOL_2_39_subckt.R_o_Q
AND_BOOL_2_wire_39_OUT:1
SEL_BOOL_2_64_subckt.OUT_Q
SEL_BOOL_2_wire_64_OUT:1
--On input:
pulse:0
```

State 7

```
--Goes to state 7:
SEL_BOOL_2_wire_64_OUT:0
SEL_INT_2_65_subckt.OUT_0_
SEL_INT_2_wire_65_OUT<0>:1
--On input:
<Unchanged>
```

State 8

```
--Goes to state 8:
OR_BOOL_4_wire_61_OUT:0
--On input:
AI_E:0
```

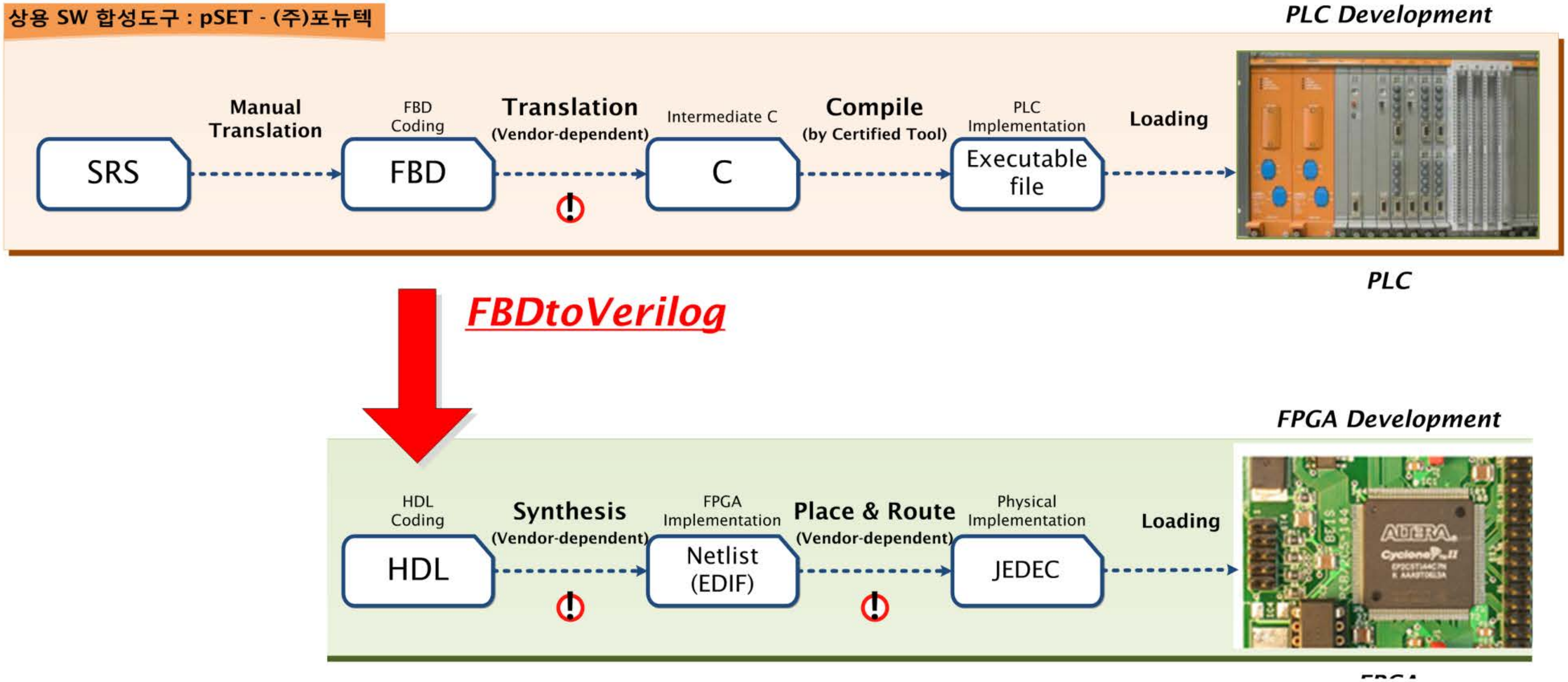
State 9

```
--Goes to state 9:
AND_BOOL_2_wire_71_OUT:0
LT_INT_2_28_subckt.R_o_Q$N
LT_INT_2_wire_28_OUT:0
OR_BOOL_4_wire_61_OUT:1
--On input:
AI_E:1
PV_OUT<1>:0
```

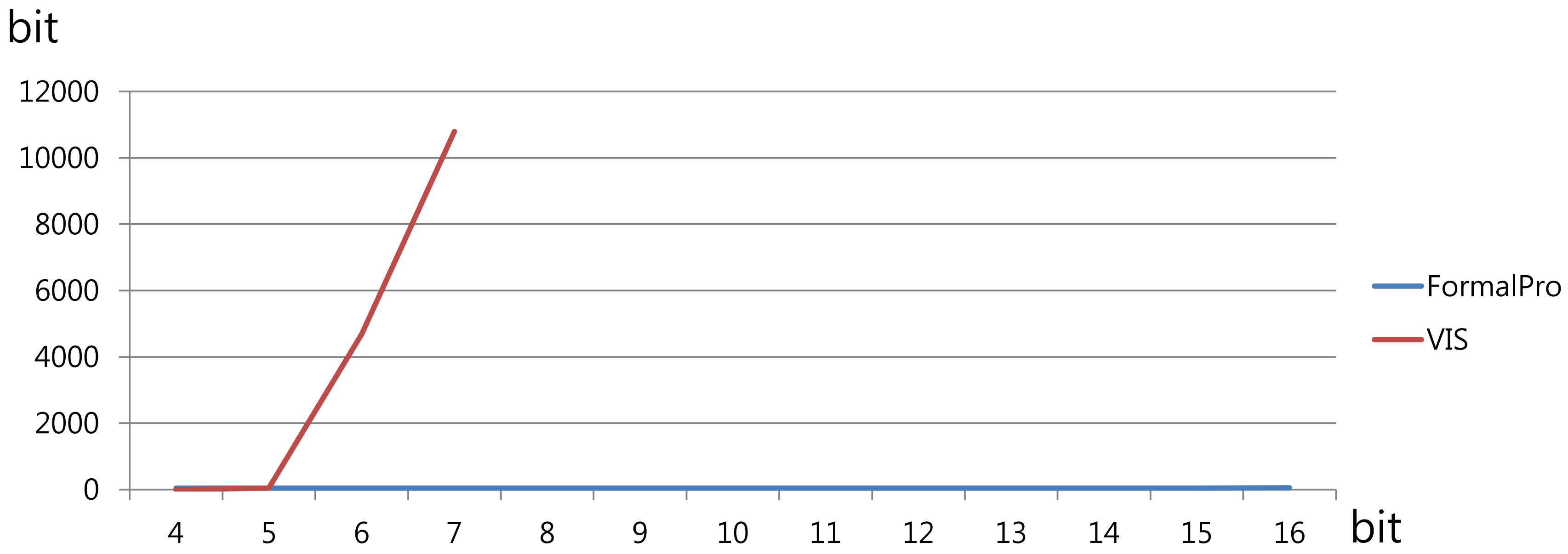
```
Networks are NOT sequentially equivalent.
```

Case study

- Bistable Process in Reactor Protection System
  - 1) FBDtoVerilog 로부터 얻은 example
  - 2) Verilog 로 작성된 example



No.	bit	Translator	# of combinational	# of pi	# of po	# of latches	# of edges	Time	Formal Pro
FIX-RISING (Trip)	4 bit	vl2mv	191	9	13	19	418	3.001 sec	OK 47.9
		EDIFtoBLIF-MV	519	9	13	19	653		
	5 bit	vl2mv	231	10	16	23	511	41.039 sec	44.8
		EDIFtoBLIF-MV	598	10	16	23	771		
	6 bit	vl2mv	274	11	19	27	607	185.288 sec (3m)	44.9
		EDIFtoBLIF-MV	696	11	19	27	902		
	7 bit	vl2mv	312	12	22	31	698	4687.821 (1h 18m)	44.8
		EDIFtoBLIF-MV	866	12	22	31	1116		
	8 bit	vl2mv	354	13	25	35	793	Over 3 h....	44.6
		EDIFtoBLIF-MV	937	13	25	35	1220		
	9 bit	vl2mv	392	14	28	39	884		45.2
		EDIFtoBLIF-MV	1102	14	28	39	1443		
	10 bit	vl2mv	432	15	31	43	977		44.7
		EDIFtoBLIF-MV	1274	15	31	43	1683		
	11 bit	vl2mv	472	16	34	47	1070		44.7
		EDIFtoBLIF-MV	1471	16	34	47	1945		
	12 bit	Vl2mv	514	17	37	51	1165		45.6
		EDIFtoBLIF-MV	1592	17	37	51	2089		
	13 bit	vl2mv	554	18	40	55	1258		45.5
		EDIFtoBLIF-MV	1735	18	40	55	2283		
	14 bit	vl2mv	594	19	43	59	1351		45.9
		EDIFtoBLIF-MV	1895	19	43	59	2506		
	15 bit	vl2mv	634	20	46	63	1444		47
		EDIFtoBLIF-MV	2091	20	46	63	2764		
	16 bit	vl2mv	675	21	49	67	1538		49.5
		EDIFtoBLIF-MV	2171	21	49	67	2893		


bit
12000
10000
8000
6000
4000
2000
0
4
5
6
7
8
9
10
11
12
13
14
15
16
bit
FormalPro
VIS

Verilog – 1

No.	Name of Logics	Translator	# of combinational	# of pi	# of po	# of latches	# of const	# of edges	Time (Reordering Option)	Etc	FomalPro
1	Hi_CNT_PRS	vl2mv	522	33	2	40	44	1224	**2.419 sec**		Ok
		EDIFtoBLIF-MV	417	33	2	40	42	558			
2	Hi_Local_Power_Density	vl2mv	17	4	2	5	5	34	**0.074 sec**		Ok
		EDIFtoBLIF-MV	43	4	2	5	7	53			
3	Hi_Log_Power	vl2mv	550	36	3	45	56	1311	**3.443 sec**		Ok
		EDIFtoBLIF-MV	472	36	3	45	47	636			
4	Hi_PZR_Pressure	vl2mv	527	[illegible]	[illegible]	40	45	1259	**3.208 sec**		Ok
		EDIFtoBLIF-MV	448	33	2	40	42	606			
5	Hi_SGL1_NR	vl2mv	525	33	2	40	47	1227	**2.260 sec**		Ok
		EDIFtoBLIF-MV	417	33	2	39	41	561			
6	Hi_SGL2_NR	vl2mv	463	33	2	40	46	1083	**2.480 sec**		Ok
		EDIFtoBLIF-MV	440	33	2	39	41	597			
7	Lo_DNBR	vl2mv	540	33	[illegible]	40	58	1314	**3.148 sec**		Ok
		EDIFtoBLIF-MV	417	[illegible]	2	40	42	560			
8	Lo_DNBR_Sta	vl2mv	17	4	2	5	5	34	**0.094 sec**		Ok
		EDIFtoBLIF-MV	43	4	2	5	7	53			
9	Lo_PZR_Pressure	vl2mv	613	13	4	68	72	1443	**Over 10 h …**	Size down 32→8	Ok
		EDIFtoBLIF-MV	1362	13	4	67	69	1914			
10	Lo_RC1_FLW	vl2mv	746	11	2	48	56	2007	**46.636 sec**	Size down 48→10	Ok
		EDIFtoBLIF-MV	1421	11	[illegible]	48	50	2010			
11	Lo_RC2_FLW	vl2mv	746	11	2	48	56	2007	**42.566 sec**	Size down 48→10	Ok
		EDIFtoBLIF-MV	1421	11	2	48	50	2010			

Negated edge

Negated edge

Negated edge

Verilog – 2

No.	Name of Logics	Translator	# of combinational	# of pi	# of po	# of latches	# of const	# of edges	Time (Reordering Option)	Etc	
12	Lo_SG1_ PRS	vl2mv	542	11	2	59	61	1279	**473.498 sec**	Size down 32→8	Ok
		EDIFtoBLIF-MV	1260	11	2	58	60	1776			
13	Lo_SG2_ PRS	vl2mv	542	11	2	59	61	1279	**589.266 sec**	Size down 32→8	Ok
		EDIFtoBLIF-MV	1260	11	2	58	60	1776			
14	Lo_SGL1_ ESF	vl2mv							**FAIL**		**Ok**
		EDIFtoBLIF-MV									
15	Lo_SGL1_ RPS	vl2mv	525	33	2	40	47	1227	**3.204 sec**		Ok
		EDIFtoBLIF-MV	457	33	2	40	42	598			
16	Lo_SGL2_ ESF	vl2mv							**FAIL**		**Ok**
		EDIFtoBLIF-MV									
17	Lo_SGL2_ RPS	vl2mv	525	33	2	40	47	1227	**3.178 sec**		Ok
		EDIFtoBLIF-MV	457	33	2	40	42	598			
18	Variable_ OverPower	vl2mv							**FAIL**		**Ok 51.4**
		EDIFtoBLIF-MV									

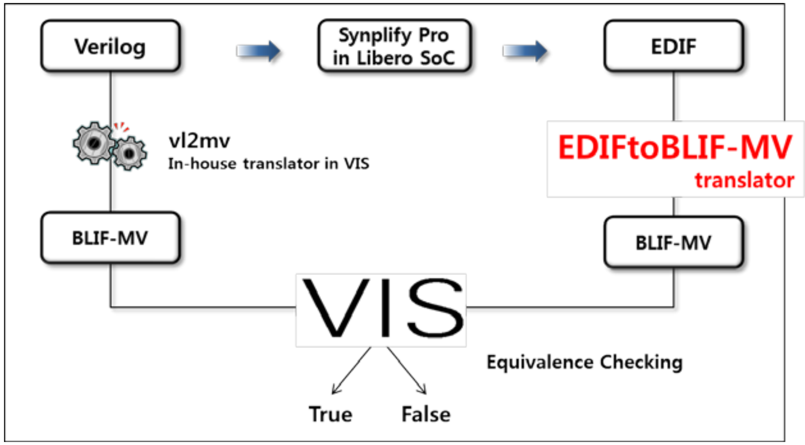
False Alarm

False Alarm

Negated edge

Conclusion and Future work

- 상용 Synthesis Tool 의 Correctness Verification Technique
  - 지원도구 EDIFtoBLIF-MV 개발
  - Case Study 수행



- Future work
  - 제시한 기법의 performance ↑
  - VIS 의 의존성 ↓

감사합니다.