A Technique for Demonstrating Safety and Correctness of Program Translators: Strategy and Case Study

Eui-Sub Kim, Junbeom Yoo
Dependable Software Laboratory
KONKUK University, Republic of Korea

Jong-Gyun Choi, Young Jun Lee, Jang-Soo Lee
Man-Machine Interface System Laboratory
Korea Atomic Energy Research Institute, Republic of Korea

2014-11-05
Contents

1. Introduction
2. The Demonstration Strategy
   - 1. Safety demonstration
   - 2. Correctness demonstration
3. The Development of Supporting Tools
   - 1. FBDtoCadenceSMV
   - 2. Scenario Generator
   - 3. FBD Simulator
   - 4. FBD-Verilog Comparator
4. Case Study
   - 1. The Safety Demonstration
   - 2. The Correctness Demonstration
5. Conclusion and Future Work

2014-11-05
1. INTRODUCTION
1. Introduction

• **Strategy and Case Study**
  – A specific translator: FBDtoVerilog
  – **Case Study**: BP (Bistable Process) of RPS (Reactor Protection system) in Nuclear Power Plants
    • It produces the **‘Shutdown’ signal** to protect a NPP from unwanted situations.

Scope

An overview of Nuclear Power Plants.
1. Introduction

- Software Development Process based on PLC
1. Introduction

• Software Development Process based on PLC

Recently, there is a trend to replace the platform from PLC to FPGA.
1. Introduction

- **PLC vs. FPGA**
  - There are differences in stages of software development process.
1. Introduction

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

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

⚠️ We must prove that the translator will work out correctly and safely
2. A DEMONSTRATION STRATEGY

1. Safety Demonstration Strategy
2. Correctness Demonstration Strategy
2. A Demonstration Strategy
2. A Demonstration Strategy

- Top goal: The translator ‘FBDtoVerilog’ should always translate source programs into target programs safely and correctly.

\[2014-11-05\]
2. A Demonstration Strategy

• Direct demonstration approach
• Indirect demonstration approach
2. A Demonstration Strategy

- Direct demonstration approach
- Indirect demonstration approach
2. A Demonstration Strategy

- Safety
  - Definition: A translator is safe, if safety properties are satisfied with the input and output programs simultaneously.

- Correctness
  - Definition: A translator is correct, if the behavior of a translated program is the same with its source program for all possible input scenarios.
2.1 The Safety Demonstration Strategy

G3 **Claim** The translator FBDtoVerilog should translate an input FBD program into a Verilog program safely

A1 Use the model checking technique

C4 The safety is defined in terms of translators

S3 **Argument** By demonstrating an FBD program and a Verilog program satisfy the same safety properties

G5 **Claim** The safety properties are appropriate

S4 **Argument** By showing safety properties are exactly reflecting the safety of an input FBD program

S5 **Argument** By showing safety properties are written in temporal logic precisely

G6 **Claim** An input FBD program satisfies the safety properties

G7 **Claim** A translated Verilog program satisfies the same properties, too

A4 An FBD program can be checked by model checking tools

A5 A Verilog program can be checked by model checking tools

A2 Safety properties are developed through safety analysis or by domain experts

A3 Safety properties are written in temporal logic for the model checker to use

G8 **Claim** Confirm whether the properties reflect the safety of target system and are well defined with formalism or not

G9 **Claim** There is a technique to enable model checking FBD programs

G10 **Claim** FBD programs satisfy the same safety properties with Verilog programs

G11 **Claim** Verilog programs satisfy the same safety properties with FBD programs

Sn 1 Inspection by experts about each domain system and formalism

Sn 2 Develop the FBDtoCadence\(\text{\textcopyright}\) \(\text{\textregistered}\) MV translator

Sn 3 Compare the model checking result of FBD programs with Verilog programs

Sn 4 Compare the model checking result of Verilog programs with FBD programs

Sn 5 Compare the model checking result of FBD programs with Verilog programs

2014-11-05
2.1 The Safety Demonstration Strategy

- Model Checking
  - Given a model of a system, exhaustively and automatically check whether this model meets a given specification.
  - We used a model checking tool CadenceSMV

A typical model checking work-flow
2.1 The Safety Demonstration Strategy

Safety Property

G5 Claim The safety properties are appropriate

G4 The safety is defined in terms of translators

G3 Claim The translator FBDtoVerilog should translate an inputted FBD program into a Verilog program safely

S3 Argument By demonstrating an FBD program and a Verilog program satisfy the same safety properties

A1 Use the model checking technique

Model Checking

G6 Claim An inputted FBD program satisfies the safety properties

G7 Claim A translated Verilog program satisfies the same properties, too

S6 Argument By satisfying the safety properties

A4 An FBD program can be checked by model checking tools

A5 A Verilog program can be checked by model checking tools

G10 Claim FBD programs satisfy the same safety properties with Verilog programs

G11 Claim Verilog programs satisfy the same safety properties with FBD programs

A8 Claim Confirm whether the properties reflect the safety of target system and are well defined with formalism or not

A7 Safety properties are developed through safety analysis or by domain experts

S5 Argument By showing safety properties are written in temporal logic precisely

S4 Argument By showing safety properties are exactly reflecting the safety of an inputted FBD program

A3 Safety properties are written in temporal logic for the model checker to use

Sn 1 Inspection by experts about each domain system and formalism

Sn 2 Develop the FBDtoCadence$ MV translator

Sn 3 Compare the model checking result of FBD programs with Verilog programs

Sn 4 Compare the model checking result of Verilog programs with FBD programs

Sn 9 Claim There is a technique to enable model checking FBD programs

Sn 9 Claim FBD programs satisfy the same safety properties with Verilog programs
2.1 The Safety Demonstration Strategy

- **Goal 5**
  - Claim the safety properties are appreciate

- **Assumption 2**
  - Safety properties are reflecting important safety features of the target input/output programs
2.1 The Safety Demonstration Strategy

- **Goal 5**
  - Claim the safety properties are appreciate

- **Assumption 2**
  - Safety properties are reflecting important safety features of the target input/output programs

- **Assumption 3**
  - They well formed with the formalism which the model checking technique requires

**Natural requirement**

"If PV_OUT (An input sensor value) is more than the TSP (Trip Set-Point) for a predefined time, then the trip signal should be fired (TRIP_LOGIC = 1) immediately."

**CTL formula**

\[ \text{AG}((PV\_OUT > TSP) \ & \ (TRIP\_CNT \geq (MAXCNT - 1)) \rightarrow \text{AX}(TRIP\_LOGIC = 1)) \]
2.1 The Safety Demonstration Strategy

- **Goal 5**
  - Claim the safety properties are appreciate

- **Assumption 2**
  - Safety properties are reflecting important safety features of the target input/output programs

- **Assumption 3**
  - They well formed with the formalism which the model checking technique requires

- **Evidence**
  - Inspection by experts about each domain system and formalism
2.1 The Safety Demonstration Strategy

- **Goal 6, 7**
  - An Inputted FBD and a translated Verilog satisfy the same porpoises

- **Assumption 4, 5**
  - Are there model checker for FBD and Verilog?
2.1 The Safety Demonstration Strategy

- **Goal 6, 7**
  - An Inputted FBD and a translated Verilog satisfy the same porpoises

- **Assumption 4, 5**
  - Are there model checker for FBD and Verilog?

- **Evidence 2**
  - Claim that there are model checkers to check both programs
    → We developed the 'FBDtoCadenceSMV'
2.1 The Safety Demonstration Strategy

- **Goal 6, 7**
  - An Inputted FBD and a translated Verilog satisfy the same porpoises

- **Assumption 4, 5**
  - Are there model checker for FBD and Verilog?

- **Evidence 2**
  - Claim that there are model checkers to check both programs
  
  → We developed the ‘FBDtoCadenceSMV’

- **Evidence 3**
  - FBD model checking result

- **Evidence 4**
  - Verilog model checking result
2.1 The Safety Demonstration Strategy

The translator work

if Safety Property & Model Checking then

G3 Claim The translator
FBDotCAD and FBDToVerilog translate
an inputted FBD program into a
Verilog program safely

G4 The safety is defined in
terms of translators

G5 Claim The safety properties
are appropriate

G6 Claim An inputted FBD
program satisfies the safety
properties

G7 Claim A translated Verilog
program satisfies the same
properties, too

S3 Argument By demonstrating
an FBD program and a Verilog
program satisfy the same
safety properties

S4 Argument By showing safety
properties are exactly
reflecting the safety of an
inputted FBD program

S5 Argument By showing safety
properties are written in
temporal logic precisely

S6 Argument By satisfying the
safety properties

A1 Use the model
checking technique

A2 Safety properties are
developed through safety
analysis or by domain
experts

A3 Safety properties are
written in temporal
logic for the model
checker to use

A4 An FBD program can
be checked by model
checking tools

A5 A Verilog program can
be checked by model
checking tools

G8 Claim Confirm whether the
properties reflect the safety
of target system and are well
defined with formalism or not

G9 Claim There is a technique to
enable model checking FBD
programs

G10 Claim FBD programs satisfy
the same safety properties
with Verilog programs

G11 Claim Verilog programs
satisfy the same safety
properties with FBD programs

Safely

if & then

The translator work

Safely

2014-11-05
2.2 The Correctness Demonstration Strategy

Correctness

Claim The translator FBDtoVerilog should translate an inputted FBD program into a Verilog program correctly

The correctness is defined in terms of translators

By demonstrating that the behavior of a Verilog program exactly corresponds to an FBD Program

Use co-simulation technique

Simulated scenarios are sufficient to cover many cases and reflect domain-specific features

Input variables are continuous in NPPs

A Verilog program can be simulated

An FBD program can be simulated

A Verilog program can be simulated

An FBD program can be simulated

Externally observable outputs of both programs can be identified

Argument By comparing both simulation results with externally observable outputs

Argument By demonstrating that the behavior of a Verilog program exactly corresponds to an FBD Program

Argument By performing the simulation of the both programs with generated scenarios

Argument By showing that scenarios can be generated as many as possible

Argument By showing that the generated scenarios reflect the real domain features

A7 Use random scenario generation technique

A8 Scenarios are generated automatically

G12 Claim Simulation scenarios are sufficient to cover many cases and reflect domain-specific features

G13 Claim The behaviors of both programs are correct regarding with the generated scenarios

G14 Claim We can generate a sufficient number of simulation scenarios

G15 Claim Domain experts can confirm that the generated scenarios reflect domain features well

G16 Claim We can simulate FBD programs reasonably

G17 Claim We can obtain the result of FBD simulation through FBD Simulator

G18 Claim We can obtain the result of the Verilog simulation through ModelSim

G19 Claim We can compare the simulation results of FBD and Verilog simultaneously

Sn 5 Develop an automatic Scenario Generator

Sn 6 Inspection by domain experts

Sn 7 Develop the FBD Simulator

Sn 8 FBD simulation results from the FBD Simulator

Sn 9 Verilog simulation result from ModelSim

Sn 10 Develop the FBD-Verilog Comparator
2.2 The Correctness Demonstration Strategy

The correctness is defined in terms of translators.

- **A7** Use random scenario generation technique
- **A8** Scenarios are generated automatically
- **A9** Input variables are continues in NPPs
- **A10** An FBD program can be simulated
- **A11** A Verilog program can be simulated
- **S11** Argument by comparing the simulation results with externally observable outputs

**Correctness**

- **G14** Claim We can generate a sufficient number of simulation scenarios
- **G15** Claim Domain experts can confirm that the generated scenarios reflect domain features well
- **G16** Claim We can simulate FBD programs reasonably
- **G17** Claim We can obtain the result of FBD simulation through FBD Simulator
- **G18** Claim We can obtain the result of the Verilog simulation through ModelSim
- **G13** Claim The behaviors of both programs are correct regarding with the generated scenarios
- **G12** Claim Simulation scenarios are sufficient to cover many cases and reflect domain-specific features
- **G4** Claim The translator FBDtoVerilog should translate an inputted FBD program into a Verilog program correctly

**Co-Simulation**

- **Sn 5** Develop an automatic Scenario Generator
- **Sn 6** Inspection by domain experts
- **Sn 7** Develop the FBD Simulator
- **Sn 8** FBD simulation results from the FBD Simulator
- **Sn 9** Verilog simulation result From ModelSim
- **Sn 10** Develop the FBD-Verilog Comparator
2.2 The Correctness Demonstration Strategy

Scenario

A7 Use random scenario generation technique

A8 Scenarios are generated automatically

A9 Input variables are continuous in NPs

A10 An FBD program can be simulated

A11 A Verilog program can be simulated

S10 Argument By performing the simulation of the both programs with generated scenarios

S11 Argument By comparing both simulation results with externally observable outputs

G12 Claim Simulation scenarios are sufficient to cover many cases and reflect domain-specific features

G13 Claim The behaviors of both programs are correct regarding with the generated scenarios

G4 Claim The translator FBDtoVerilog should translate an inputted FBD program into a Verilog program correctly

G5 The correctness is defined in terms of translators

G7 Argument By demonstrating that the behavior of a Verilog program exactly corresponds to an FBD Program

G6 Use co-simulation technique

Co-Simulation

G14 Claim We can generate a sufficient number of simulation scenarios

G15 Claim Domain experts can confirm that the generated scenarios reflect domain features well

G16 Claim We can simulate FBD programs reasonably

G17 Claim We can obtain the result of FBD simulation through FBD Simulator

G18 Claim We can obtain the result of the Verilog simulation through ModelSim

G19 Claim We can compare the simulation results of FBD and Verilog simultaneously

Sn 5 Develop an automatic Scenario Generator

Sn 6 Inspection by domain experts

Sn 7 Develop the FBD Simulator

Sn 8 FBD simulation results from the FBD Simulator

Sn 9 Verilog simulation result from ModelSim

Sn 10 Develop the FBD-Verilog Comparator
The translator work Correctly if & then

if Scenarios & Co-Simulation then

2.2 The Correctness Demonstration Strategy

- Scenarios
- Co-Simulation
- Correctly
- Argument By demonstrating that the behavior of a Verilog program correctly corresponds to an FBD Program
- The correctness is defined in terms of translators
- The behaviors of both programs are correct regarding with the generated scenarios
- An FBD program can be simulated
- Argument By comparing both simulation results with externally observable outputs
- Argument By performing the simulation of the both programs with generated scenarios
- Argument By showing that the generated scenarios reflect the real domain features
- Argument By showing that scenarios can be generated as many as possible
- Scenarios are generated automatically
- Use random scenario generation technique

G14 Claim We can generate a sufficient number of simulation scenarios
G15 Claim Domain experts can confirm that the generated scenarios reflect domain features well
G5 The correctness is defined in terms of translators
G12 Simulation scenarios are sufficient to cover many cases and reflect domain specific features
S8 Argument By showing that scenarios can be generated as many as possible
S9 Argument by showing that the generated scenarios reflect the real domain features
S10 Argument By performing the simulation of the both programs with generated scenarios
A12 Externally observable outputs of both programs can be identified
A10 An FBD program can be simulated
A9 Initial variables are continues in NPS
A8 Scenarios are generated automatically
A7 Use random scenario generation technique
S4 Claim The translator FBD programs should simulate a Verilog program correctly
S7 Argument By demonstrating that the behavior of a Verilog program exactly corresponds to an FBD Program
A6 Use co-simulation technique
In summary

- In summary, we constructed our purpose and strategy with the GSN.
  - We first set up the top-level goal (G1) and divided it into two parts, safety (G3) and correctness (G4), then presented sub-goals, arguments and evidences to accomplish upper goals.

(G1) ‘FBDtoVerilog’ does work safely and correctly

All evidences are well founded
1. FBDtoCadenceSMV
2. Scenario Generator
3. FBD Simulator
4. FBD-Verilog Comparator

3. THE DEVELOPMENT OF SUPPORTING TOOLS
3.1 FBDtoCadenceSMV

- FBD program → input program of Cadence SMV (Model checker)
3.2 Scenario Generator

- The ‘Scenario Generator’ randomly generates a number of scenarios within predefined constraints on input values.
3.3 FBD Simulator

- The ‘FBD Simulator’ works in two modes.
  - This FBD Simulator executes one scenario and visualizes the results in a form of graphical chart.
  - It support a verification of functionality of FBD.
3.4 FBD-Verilog Comparator

- Automatic comparison between FBD simulation and Verilog simulation results.
4. CASE STUDY

1. The Safety Demonstration
2. The Correctness Demonstration
4. Case Study

- KNIC project RPS (Reactor Protection System) BP (Bistable Process)

**FBD** program for PLC

**Verilog** program for FPGA

Correct?
4.1 The Safety Demonstration (G3)

- We performed model checking with the Cadence SMV

- We developed 28 safety properties with assistant from domain exports and referable papers.

- Ex)
  - "If PV_OUT (An input sensor value) is more than the TSP (Trip Set-Point) for a predefined time, then the trip signal should be fired (TRIP_LOGIC = 1) immediately."

  - : \( AG((PV\_OUT > TSP) \& (TRIP\_CNT >= (MAXCNT - 1)) \rightarrow AX(TRIP\_LOGIC = 1)) \)
FBD program for PLC

FBDtoVerilog

Verilog program for FPGA

Safety Properties

True

False

Counter Example

Cadence SMV

True

False

Counter Example

Cadence SMV
Model checking results

[AG (((FV_OUT>TSP) & (TRIP_LOGIC=0)) & (TRIP_CNT=0)) \rightarrow (AX (TRIP_CNT=1))] true

[AG (((FV_OUT>TSP) & (TRIP_CNT=0)) & (TRIP_CNT=0)) \rightarrow (AX (TRIP_CNT=1))] true

[AG (((FV_OUT>TSP) & (TRIP_CNT=0)) & (TSP=20)) \rightarrow (AX (TSP=20))] true

[AG (((FV_OUT<TSP) & (TRIP_LOGIC=1)) & (TRIP_CNT=20)) \rightarrow (AX (TRIP_CNT=1))] true

[AG (((FV_OUT<TSP) & (TRIP_CNT=1)) & (TSP=15)) \rightarrow (AX (TRIP_CNT=20))] true

[AG (((FV_OUT<TSP) & (TRIP_CNT=0)) & (FIRIP_CNT=5)) \rightarrow (AX (FIRIP_CNT=0))] true

Set file "FIX-RISING.warn" for warnings.

user time.......................... 0.0936006 s
system time........................ 0.0156001 s

Resources used

---
4.2 The Correctness Demonstration (G4)

- We performed co-simulation with co-simulation environment.

Co-Simulation Environment
### 4.2 The Correctness Demonstration (G4)

<table>
<thead>
<tr>
<th>Name of Logic</th>
<th>Scenarios</th>
<th>Initial Values</th>
<th>Rate of Change</th>
<th>Cycles</th>
</tr>
</thead>
<tbody>
<tr>
<td>FIX-RISING</td>
<td>10,000</td>
<td>27,000 - 28,000 (Stepwise: 100)</td>
<td>10 - 100 (Stepwise: 10)</td>
<td>100</td>
</tr>
<tr>
<td>FIX-FALLING</td>
<td>10,000</td>
<td>12,000 - 13,000 (Stepwise: 100)</td>
<td>10 - 100 (Stepwise: 10)</td>
<td>100</td>
</tr>
</tbody>
</table>
5. CONCLUSION AND FUTURE WORK
5. Conclusion

- This paper proposed an indirect strategy for demonstrating the safety and correctness of the ‘FBDtoVerilog’ translator.

- We used the safety case technique and GSN to explain the proposed strategy more precisely and systematically.

- We also developed several CASE tools to support for deriving evidences.
  - ‘FBDtoCadenceSMV’, ‘Scenario Generator’, ‘FBD Simulator’ and ‘FBD-Verilog Comparator’.

- We then performed a case study with an FBD program of the KNICS APR-1400 RPS BP in order to demonstrate the safety and correctness of the ‘FBDtoVerilog’ indirectly, according to the demonstration strategy proposed.
Future work

• We are now trying to increase the confidence and thoroughness of the ‘Scenario Generator’.

• We are also planning to apply to other translators which we developed, such as ‘FBDtoC’ and ‘NuSCRtoFBD’.

• We expect to extend the proposed techniques into a safety and correctness demonstration framework for general translators and compilers.
Thank you for your attention ...

Contact : atang34@konkuk.ac.kr (Eui-Sub Kim)