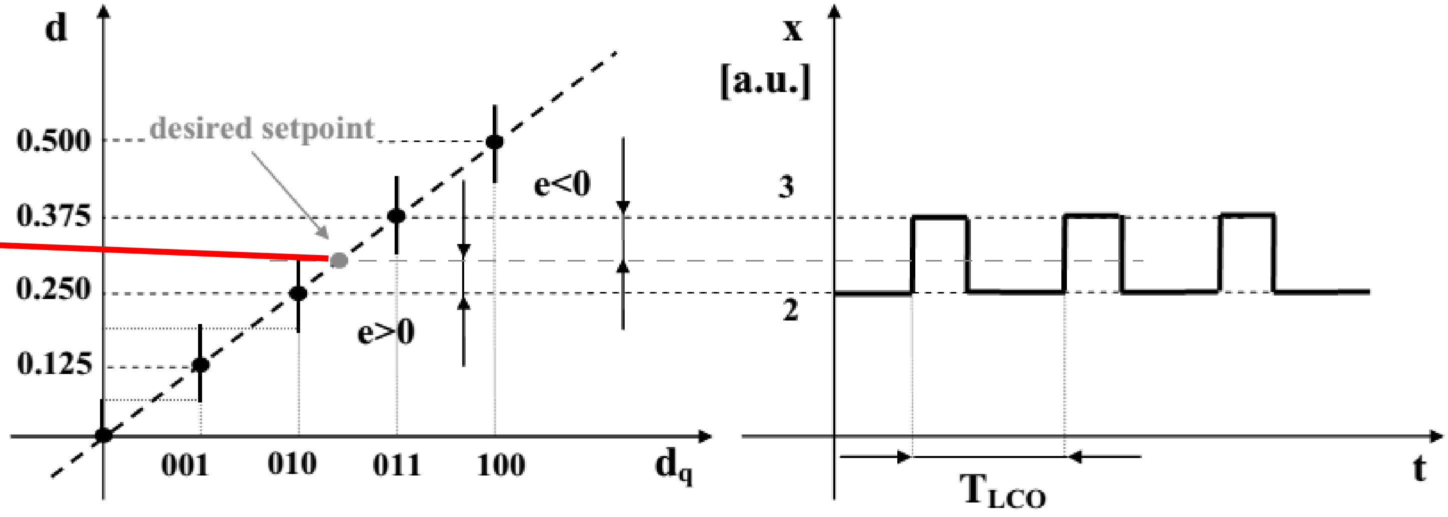
SMT-Based Bounded Model Checking of Fixed-Point Digital Controllers

Iury V. Bessa, Renato B. Abreu, Lucas C. Cordeiro, and João E. C. Filho
iurybessa@ufam.edu.br

Application of a Digital Controller to a Power DC-DC Converter

- Digital controllers have become pervasive in power eletronics applications
- Despite several advantages, they present some limitations for these applications

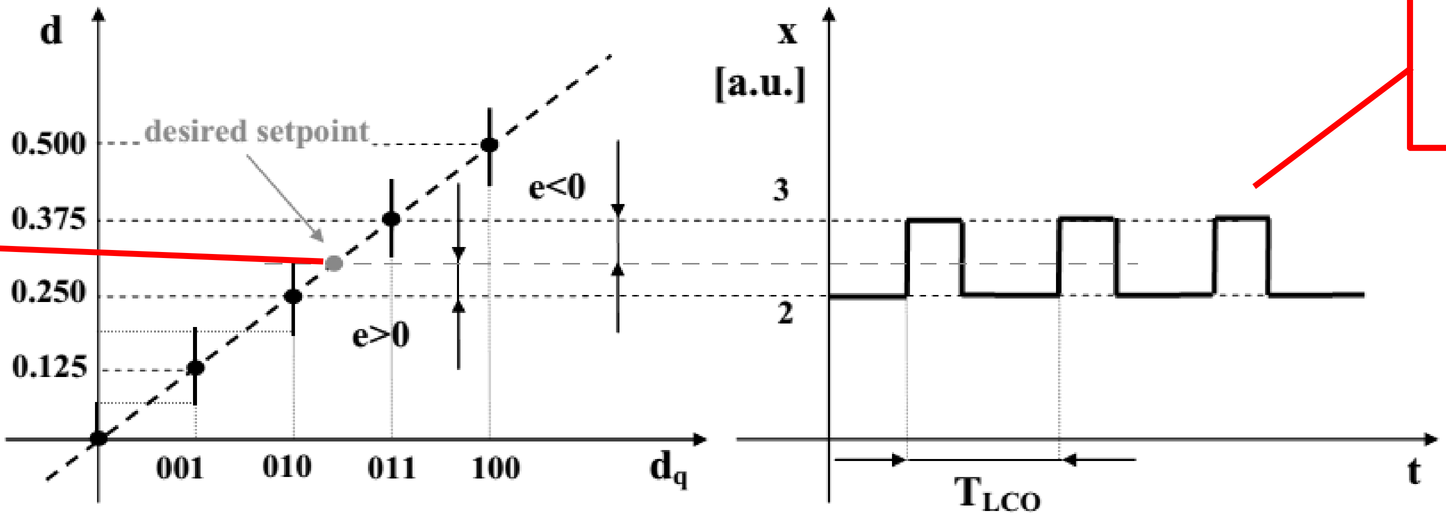
The desired setpoint may not be a representable value due to the quantization effects



Application of a Digital Controller to a Power DC-DC Converter

- Limit Cycle (LC) oscillations require high effort from engineers
- Round-off errors in products or overflows in sums may cause oscillations
- The output voltage might present an undesireble oscillation

The desired setpoint may not be a representable value due to the quantization effects

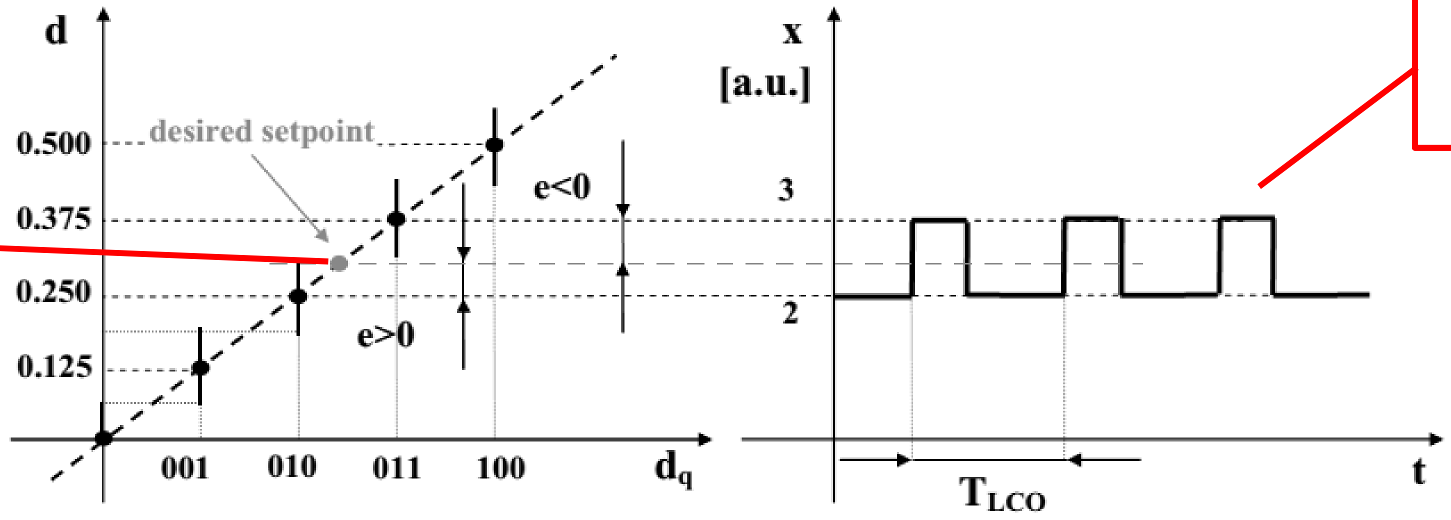


Limit cycle oscillations

Application of a Digital Controller to a Power DC-DC Converter

- More **energy losses** and **shorter silicon lifespan**
- LC's are actually verified trough time-domain simulations
  - This is na inefficient method since it is time-consuming and not conclusive

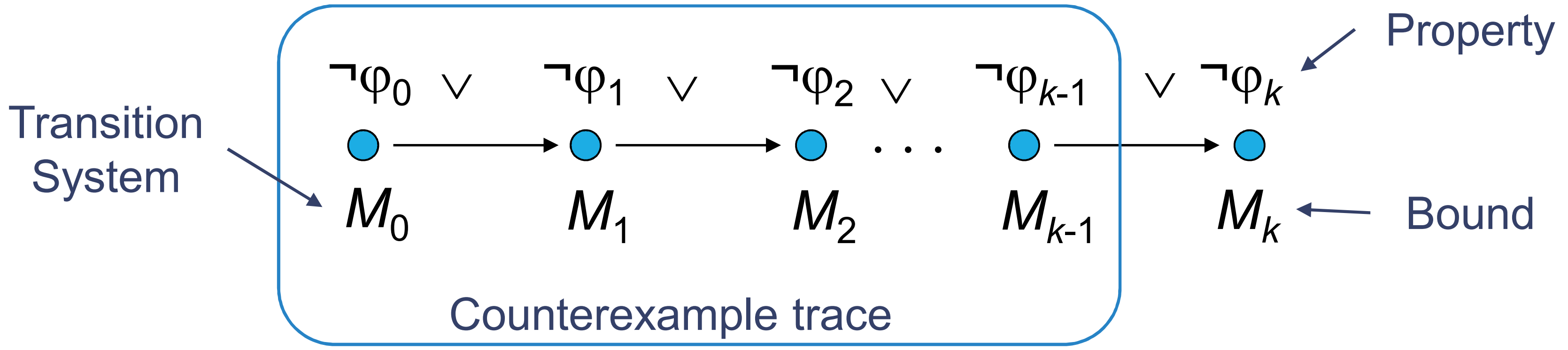
The desired setpoint may not be a representable value due to the quantization effects



Limit cycle oscillations

Bounded Model Check (BMC)

- Basic Idea: given a transition system *M*, check negation of a given property *φ* up to given depth *k*



- Translated into a VC ψ such that: **ψ is satisfiable iff φ has counterexample of max. depth k**
- BMC has been applied successfully to verify (embedded) software since early 2000's, but it has not been used to verify digital controllers

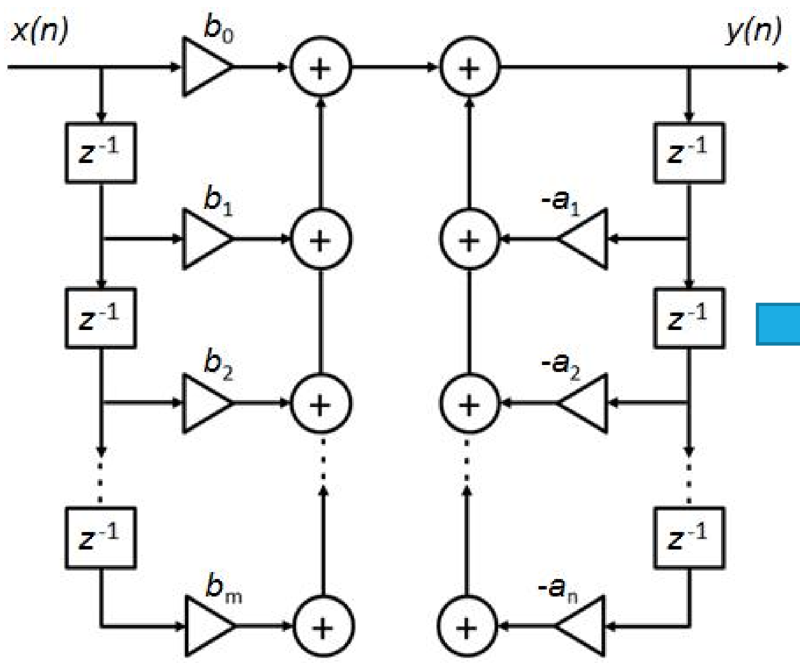
Objectives of this work

Perform bounded model checking of digital controllers implemented in direct forms

- Investigate the FWL effects in fixed-point digital controllers implementation via a BMC tool
- Propose a methodology for digital controllers implementation with the aid of a BMC tool: the DCVerifier
- Verification engine used: ESBMC (*Efficient SMT-based Context-Bounded Model Checker*)
- Properties to be verified:
  - Overflows
  - Limit Cycles
  - Time Contraints
  - Stability

Digital Controllers Implementation Forms

- Digital controllers implementation forms:
  - Direct form
  - Companion form
  - Jordan form
  - Diagonal form
  - Ladder form
  - Delta form
- Direct Forms
  - DFI
  - DFII
  - DTFII




```
float controller()
{
     float yn=0;
     for (int k=0; k<M; k++)
     {
          yn += *b++ * *x--;
     }
     for (int k=1; k<N; k++)
     {
          yn-= *a++ * *y--;
     }
     return yn;
}
```

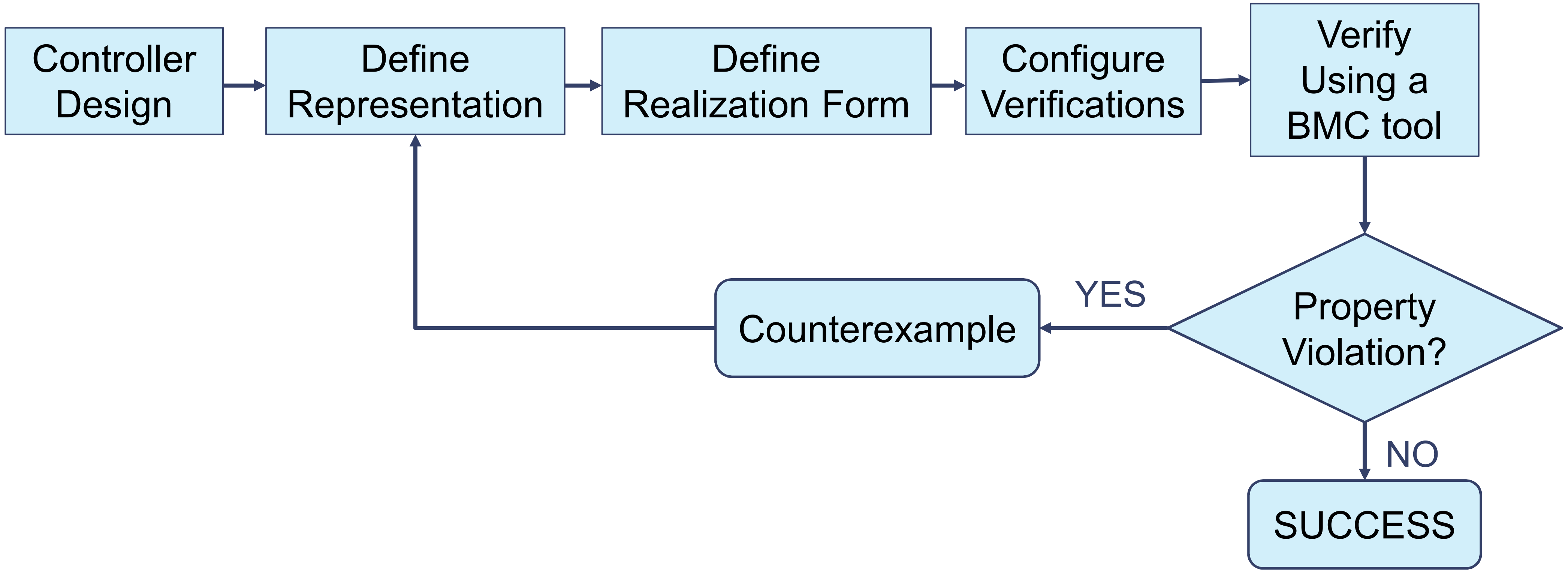
Digital Controllers Implementation Aspects

- Reduced dynamical range
- Quantization effects (FWL):
  - **Overflows:** occurs when a sum or product exceeds the maximum representable value
  - **Limit Cycles:** oscillations in output that keep a constant input due to round-offs and overflows
  - **Output errors:** the response presents deviations from the expected value
- Time constraints
- Coefficients round-off:
  - Poles and zeros sensitivity: dynamical behavior changes
  - Stability issue

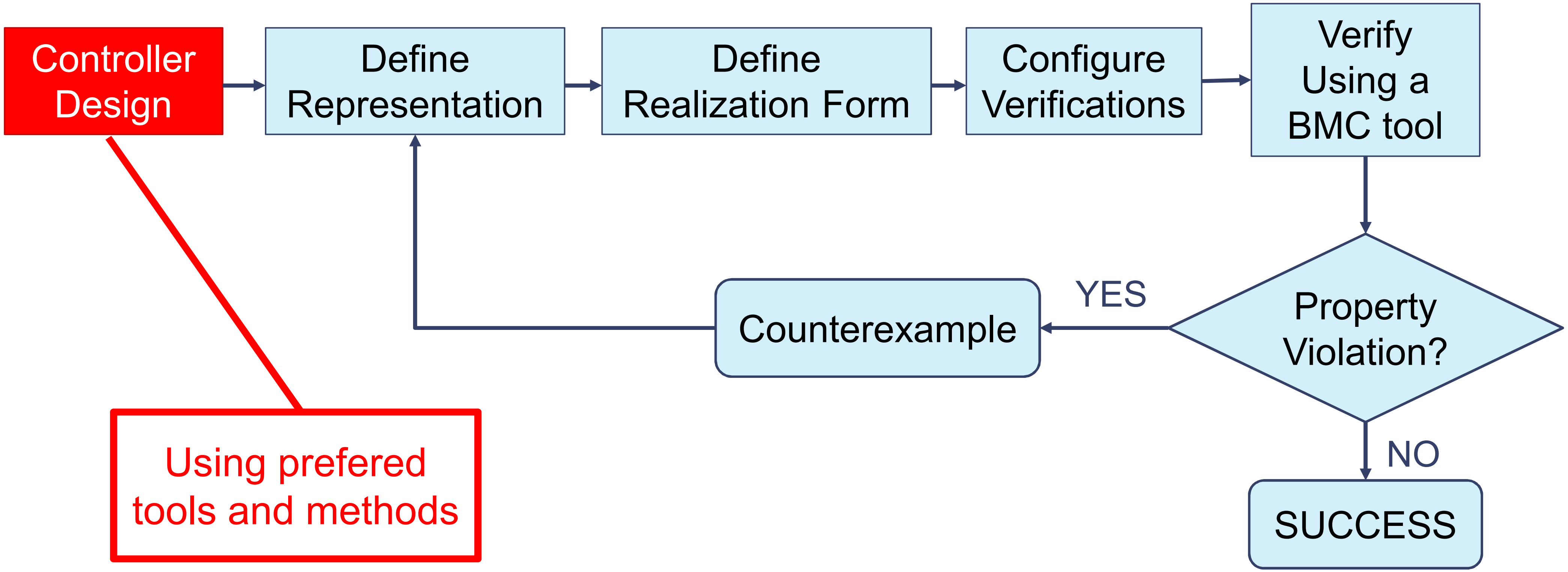
Digital Controllers Verification Paradigm

- Techniques in order to avoid problems:
  - **Scaling:** may prevent overflows, but enhances the output error
  - **Resolution changes** (number of bits): boosts the precision, reducing errors and preventing LC
  - **Linear and non-linear compesations:** an aditional control loop may rectify the LCs
  - **Non-fragile Control:** the deviations of FWL effects are considered in design as uncertains, and the designed controller should be robust to them
- Digital controllers implementation validation:
  - Based on simulations and tests
  - Consume a lot of effort and time
  - Cannot cover all the possibilities

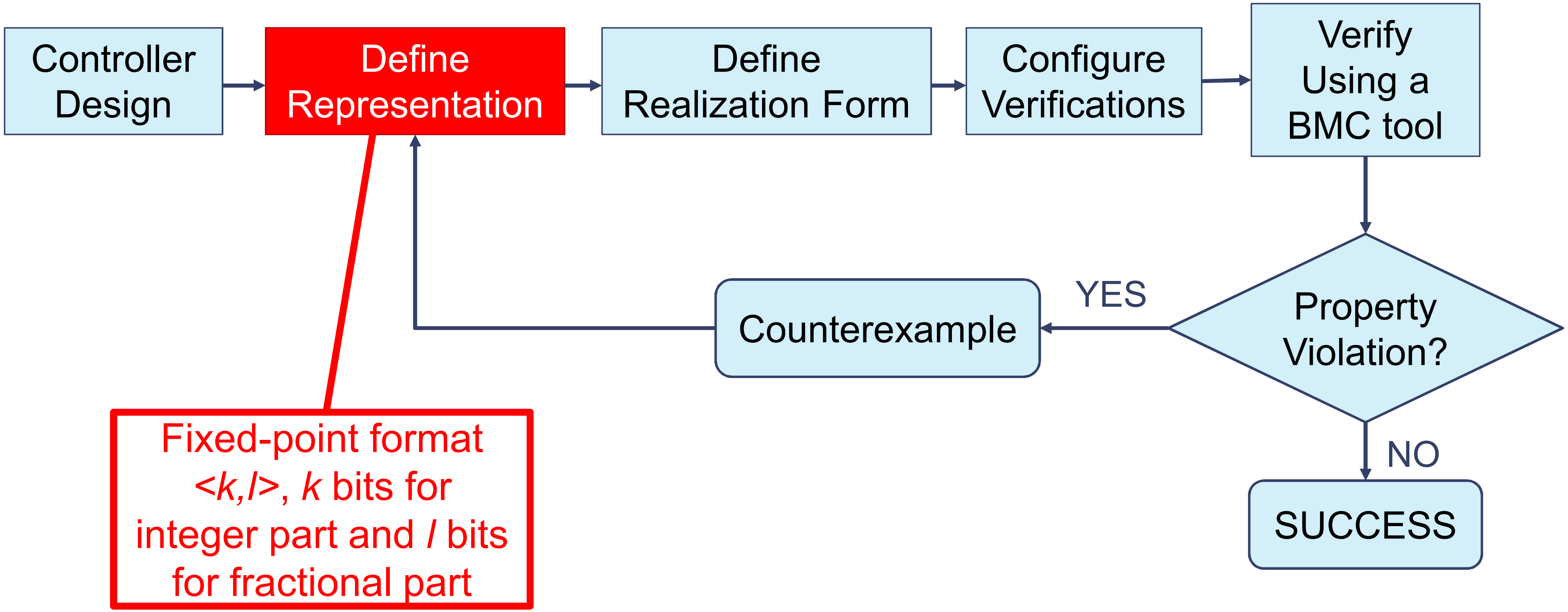
DCVerifier: Digital Controllers Implementation with Bounded Model Checking



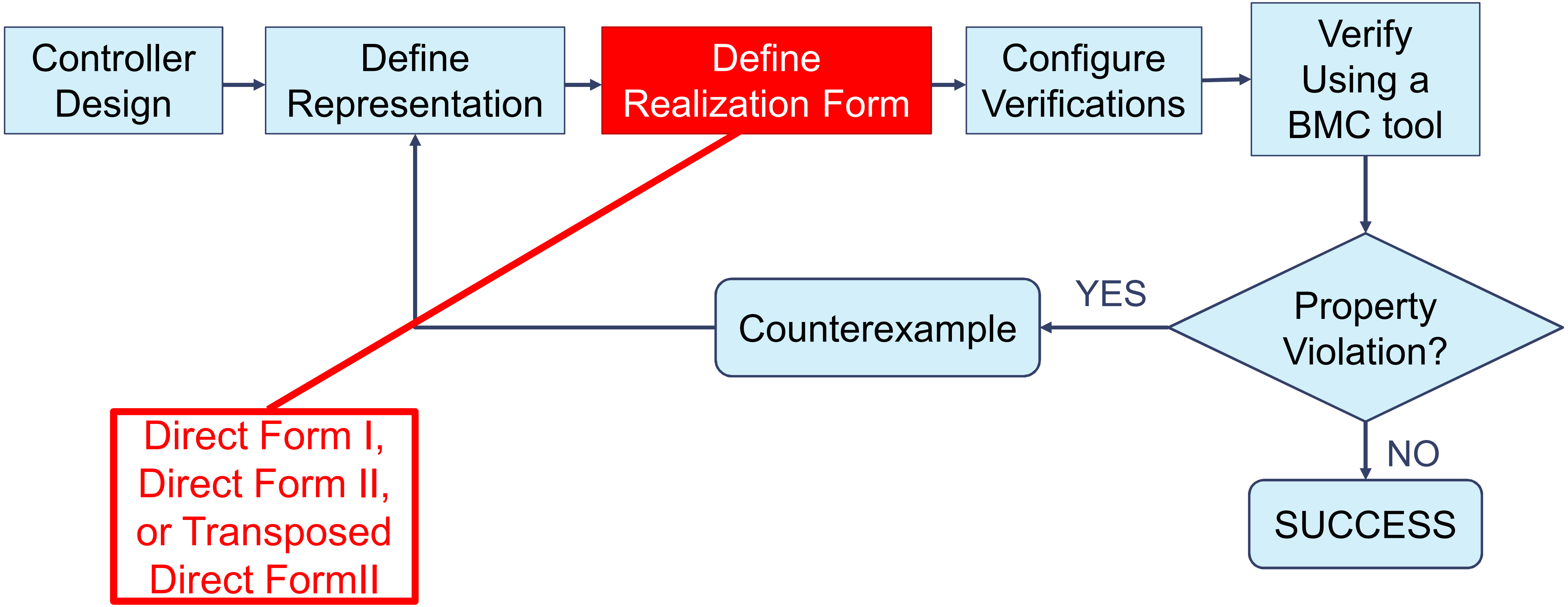
DCVerifier: Digital Controllers Implementation with Bounded Model Checking



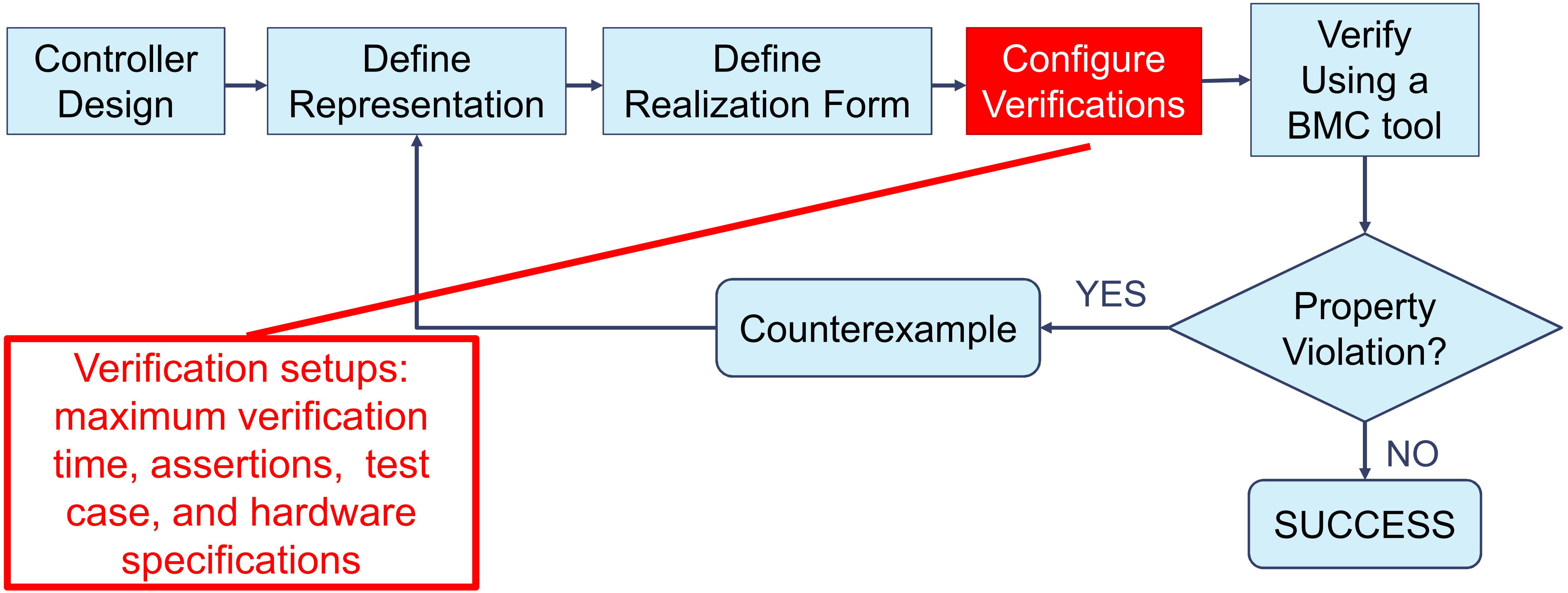
DCVerifier: Digital Controllers Implementation with Bounded Model Checking



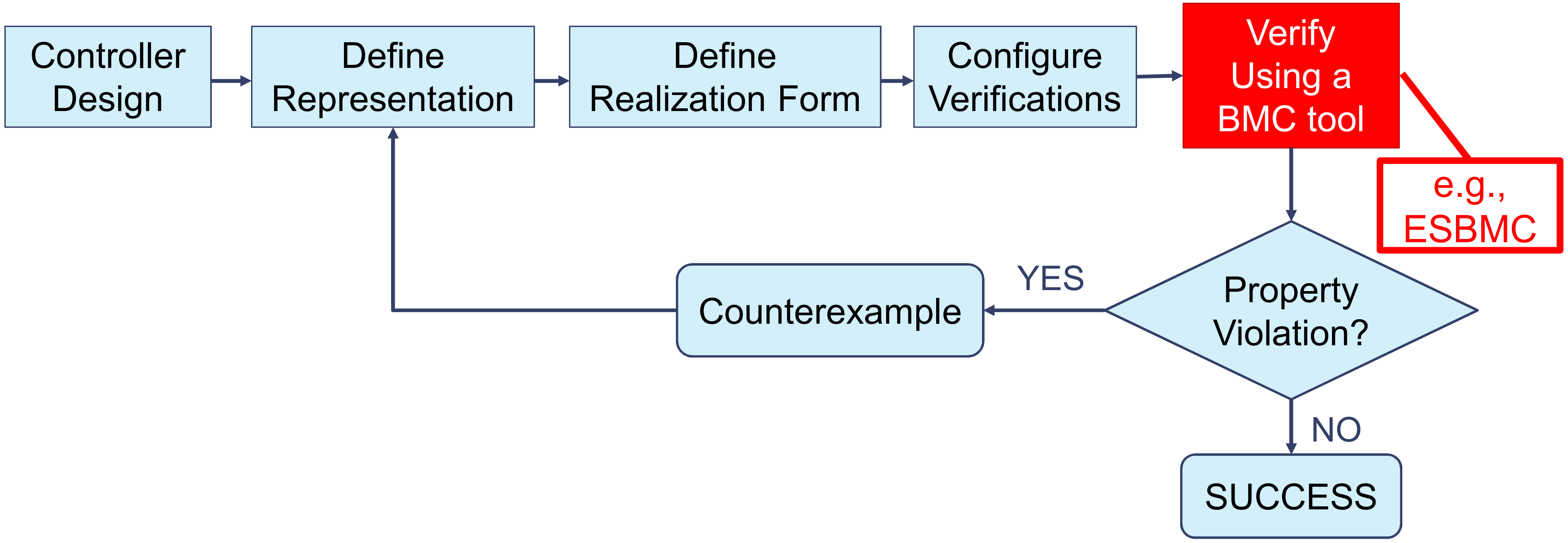
DCVerifier: Digital Controllers Implementation with Bounded Model Checking



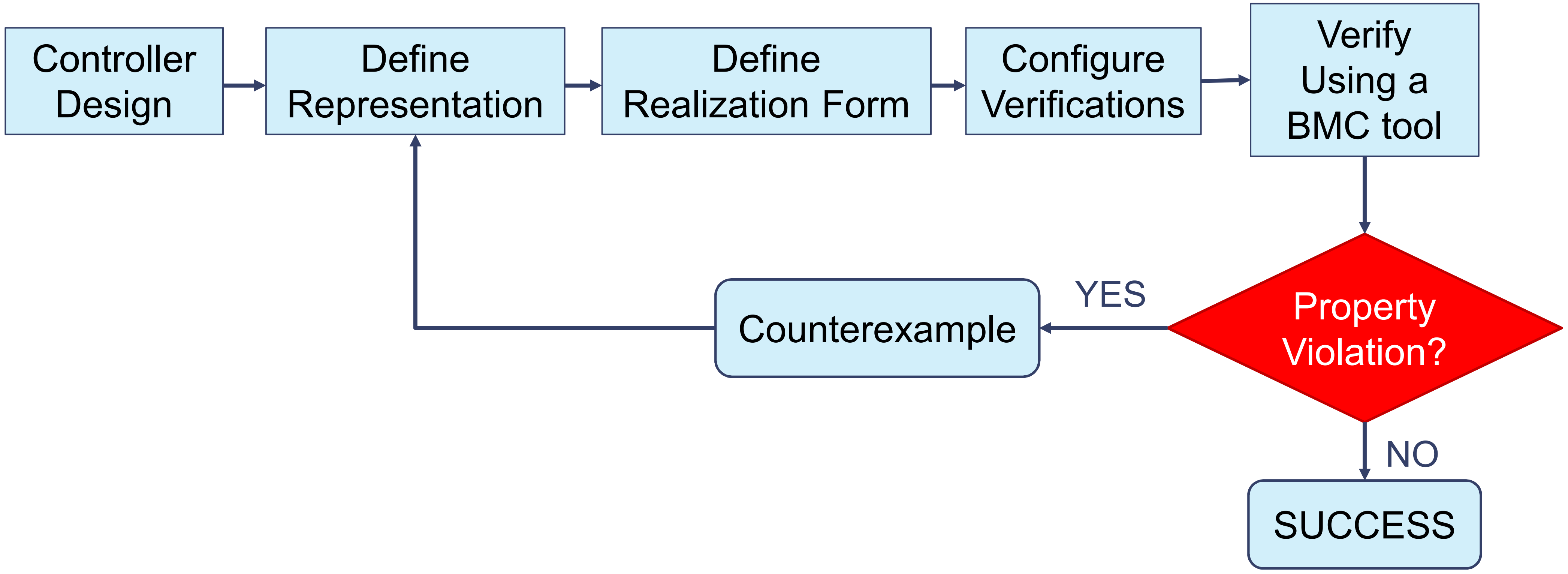
DCVerifier: Digital Controllers Implementation with Bounded Model Checking



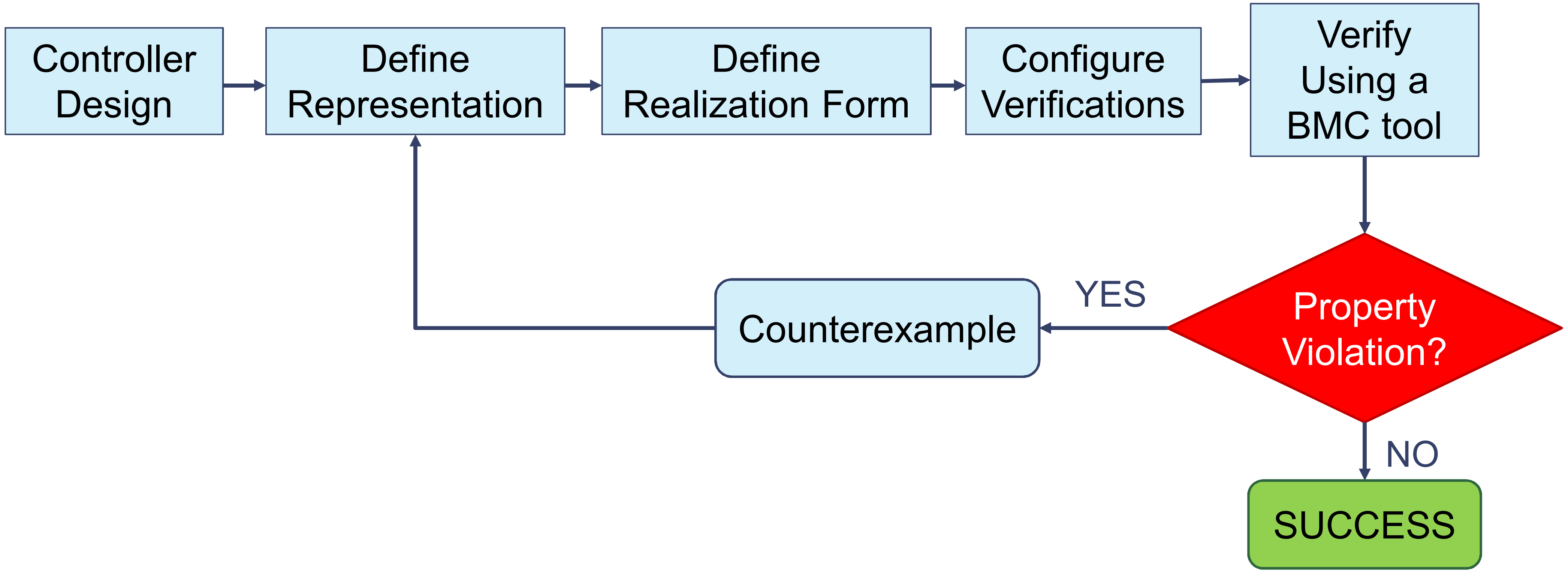
DCVerifier: Digital Controllers Implementation with Bounded Model Checking



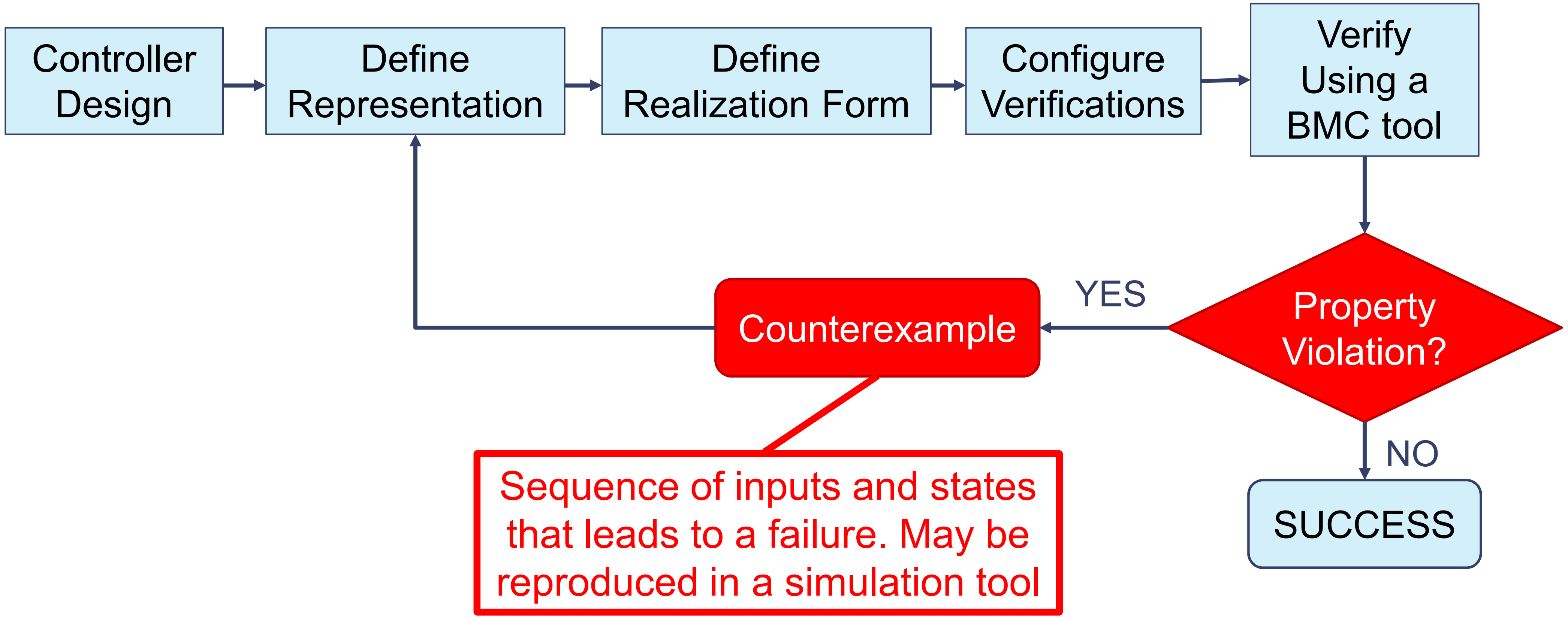
DCVerifier: Digital Controllers Implementation with Bounded Model Checking



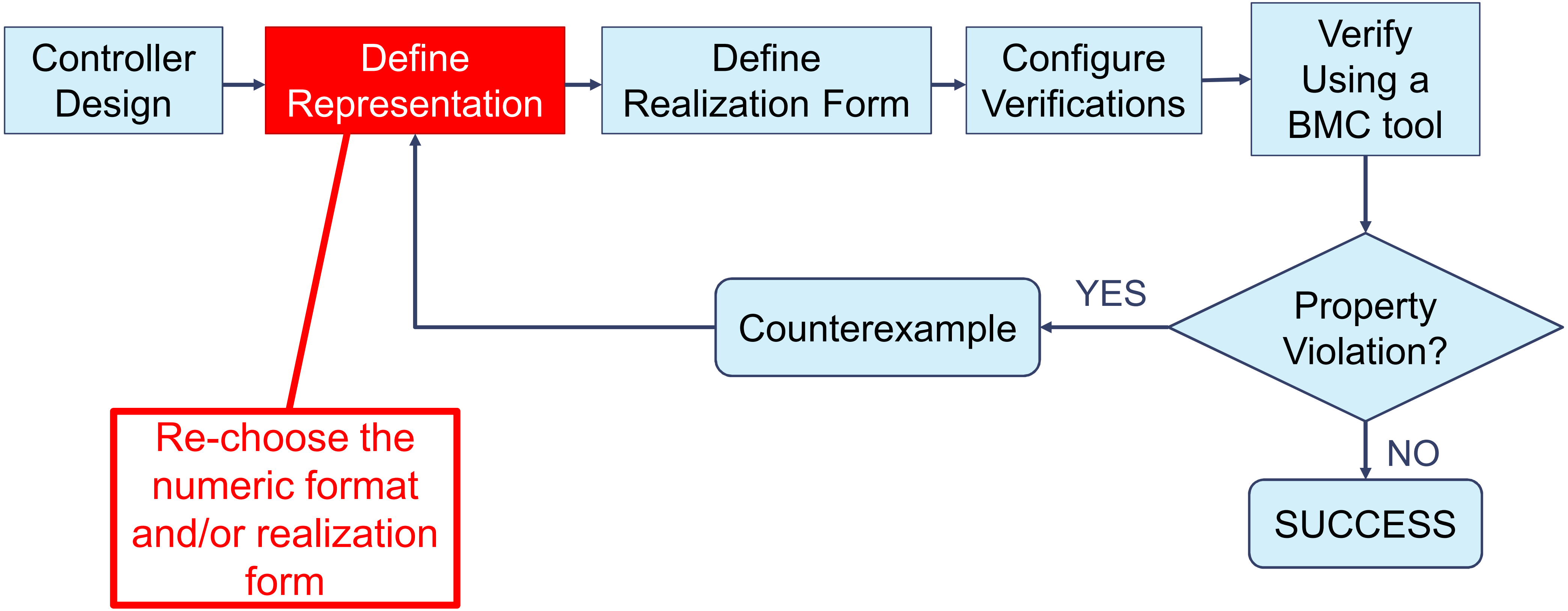
DCVerifier: Digital Controllers Implementation with Bounded Model Checking



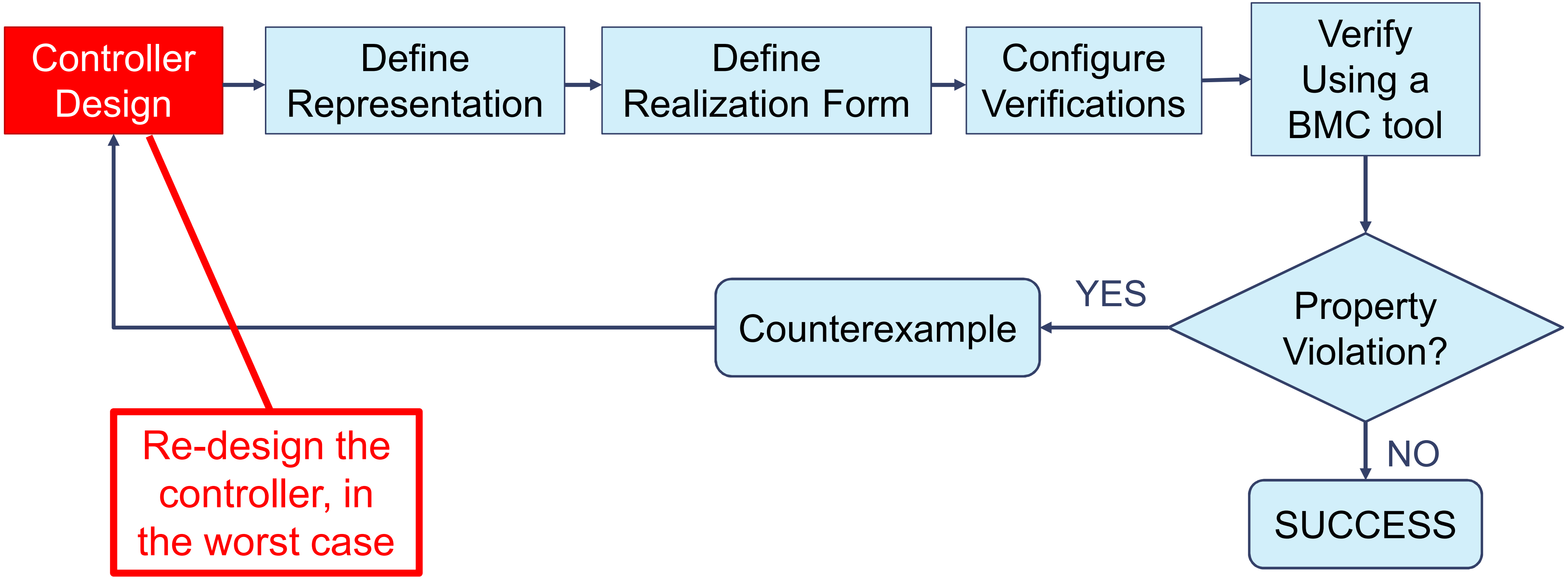
DCVerifier: Digital Controllers Implementation with Bounded Model Checking



DCVerifier: Digital Controllers Implementation with Bounded Model Checking



DCVerifier: Digital Controllers Implementation with Bounded Model Checking



DCVerifier usage example

Controller Design → Define Representa tion → Define Realization Form → Configure Verification → Verify using a BMC Tool → Result

DCVerifier usage example

Controller Design → Define Representation → Define Realization Form → Configure Verification → Verify using a BMC Tool → Result

- $C(z) = \frac{0.2(z^2-2z+1)}{z^2-0.25}$

DCVerifier usage example

Controller Design → Define Representation → Define Realization Form → Configure Verification → Verify using a BMC Tool → Result

- $C(z) = \frac{0.2(z^2-2z+1)}{z^2-0.25}$

- $<3,12>$:3 bits for integer part and 12 bits for fractional part
- Dynamical Range: [-1,1]

Numeric format choosen based on impulse response sum and in the hardware limitations

DCVerifier usage example

Controller Design → Define Representation → Define Realization Form → Configure Verification → Verify using a BMC Tool → Result

- $C(z) = \frac{0.2(z^2-2z+1)}{z^2-0.25}$

- $< 3,12 >$:3 bits for integer part and 12 bits for fractional part
- Dynamical Range: [-1,1]

- DFII

Random first trial

DCVerifier usage example

Controller Design	Define Representation	Define Realization Form	Configure Verification	Verify using a BMC Tool	Result
• $C(z) = \frac{0.2(z^2-2z+1)}{z^2-0.25}$	• $<3,12>$:3 bits for integer part and 12 bits for fractional part • Dynamical Range: [-1,1]	• DFII	• Verify overflow • Verification time:3600 • MSP430 16-bit 16 MHz		

DCVerifier usage example

Controller Design

- $C(z) = \frac{0.2(z^2-2z+1)}{z^2-0.25}$

Define Representation

- $< 3,12 >$:3 bits for integer part and 12 bits for fractional part
- Dynamical Range: [-1,1]

Define Realization Form

- DFII

Configure Verification

- Verify overflow
- Verification time:3600
- MSP430 16-bit 16 MHz

Verify using a BMC Tool

- ESBMC
- Bound: 10 samples

Result

DCVerifier usage example

Controller Design

- $C(z) = \frac{0.2(z^2-2z+1)}{z^2-0.25}$

Define Representation

- $<3,12>$:3 bits for integer part and 12 bits for fractional part
- Dynamical Range: [-1,1]

Define Realization Form

- DFII

Configure Verification

- Verify overflow
- Verification time:3600
- MSP430 16-bit 16 MHz

Verify using a BMC Tool

- ESBMC
- Bound: 10 samples

Result

- Verification Failed

Failure due to a sum overflow (sum result = 2.0879 > 1).
Input sequence: {0.9995, -0.9995, 0.9995, 1, 1, 1, 0.9995, 0.9995, 0.9995, 0.9995, 1}
Redefine the implementation!

DCVerifier usage example

Controller Design → **Define Representation** → Define Realization Form → Configure Verification → Verify using a BMC Tool → Result

- $C(z) = \frac{0.2(z^2-2z+1)}{z^2-0.25}$

- $< 3,12 >$:3 bits for integer part and 12 bits for fractional part
- Dynamical Range: [-1,1]

Maintain the Representation

DCVerifier usage example

Controller Design → Define Representation → **Define Realization Form** → Configure Verification → Verify using a BMC Tool → Result

- $C(z) = \frac{0.2(z^2-2z+1)}{z^2-0.25}$

- $< 3,12 >$:3 bits for integer part and 12 bits for fractional part
- Dynamical Range: [-1,1]

- TDFII

Change the Realization Form
TDFII presents less sums and products

DCVerifier usage example

Controller Design → Define Representation → Define Realization Form → Configure Verification → Verify using a BMC Tool → Result

Controller Design
- $C(z) = \frac{0.2(z^2-2z+1)}{z^2-0.25}$

Define Representation
- $<3,12>$:3 bits for integer part and 12 bits for fractional part
- Dynamical Range: [-1,1]

Define Realization Form
- TDFII

Configure Verification
- Verify overflow
- Verification time:3600
- MSP430 16-bit 16 MHz

DCVerifier usage example

Controller Design	Define Representation	Define Realization Form	Configure Verification	Verify using a BMC Tool	Result
• $C(z) = \frac{0.2(z^2-2z+1)}{z^2-0.25}$	• $< 3,12 >$:3 bits for integer part and 12 bits for fractional part • Dynamical Range: [-1,1]	• TDFII	• Verify overflow • Verification time:3600 • MSP430 16-bit 16 MHz	• ESBMC • Bound: 10 samples	

DCVerifier usage example

Controller Design	Define Representation	Define Realization Form	Configure Verification	Verify using a BMC Tool	Result
• $C(z) = \frac{0.2(z^2-2z+1)}{z^2-0.25}$	• $< 3{,}12 >$:3 bits for integer part and 12 bits for fractional part • Dynamical Range: [-1,1]	• TDFII	• Verify overflow • Verification time:3600 • MSP430 16-bit 16 MHz	• ESBMC • Bound: 10 samples	

Repeat the test

DCVerifier usage example

Controller Design → Define Representation → Define Realization Form → Configure Verification → Verify using a BMC Tool → Result

Controller Design
- $C(z) = \frac{0.2(z^2-2z+1)}{z^2-0.25}$

Define Representation
- $< 3,12 >$:3 bits for integer part and 12 bits for fractional part
- Dynamical Range: [-1,1]

Define Realization Form
- TDFII

Configure Verification
- Verify overflow
- Verification time:3600
- MSP430 16-bit 16 MHz

Verify using a BMC Tool
- ESBMC
- Bound: 10 samples

Result
- SUCESS

The problem was solved

DCVerifier usage example

Controller Design → Define Representation → Define Realization Form → Configure Verification → Verify using a BMC Tool → Result

Controller Design

- $C(z) = \frac{0.2(z^2-2z+1)}{z^2-0.25}$

Define Representation

- $< 3,12 >$:3 bits for integer part and 12 bits for fractional part
- Dynamical Range: [-1,1]

Define Realization Form

- TDFII

Configure Verification

- Verify limit cycle
- Verification time:3600
- MSP430 16-bit 16 MHz

Verify using a BMC Tool

- ESBMC
- Bound: 10 samples

But verifing limit cycles...

DCVerifier usage example

Controller Design	Define Representation	Define Realization Form	Configure Verification	Verify using a BMC Tool	Result
• $C(z) = \frac{0.2(z^2-2z+1)}{z^2-0.25}$	• $<3,12>$:3 bits for integer part and 15 bits for fractional part • Dynamical Range: [-1,1]	• TDFII	• Verify limit cycle • Verification time:3600 • MSP430 16-bit 16 MHz	• ESBMC • Bound: 10 samples	• Verification Failed

Appears an oscillation: {-0.002, -0.002, -0.0015, -0.0015, -0.002, -0.002, -0.0015, -0.0015, -0.002, -0.002}.

Zero input sequence

Redefine the implementation!

DCVerifier usage example

Controller Design → Define Representation → Define Realization Form → Configure Verification → Verify using a BMC Tool → Result

- **Controller Design**
  - $C(z) = \frac{0.2(z^2-2z+1)}{z^2-0.25}$
- **Define Representation**
  - $< 3,4 >$:3 bits for integer part and 4 bits for fractional part
  - Dynamical Range: [-1,1]
- **Define Realization Form**
  - TDFII
- **Configure Verification**
  - Verify limit cycle
  - Verification time:3600
  - MSP430 16-bit 16 MHz
- **Verify using a BMC Tool**
  - ESBMC
  - Bound: 10 samples

Verifing with a different representation...
There is a trade off: the oscillation is solved; however there is an accurate loss.

DCVerifier usage example

Controller Design
- $C(z) = \frac{0.2(z^2-2z+1)}{z^2-0.25}$

Define Representation
- $<3,4>$:3 bits for integer part and 4 bits for fractional part
- Dynamical Range: [-1,1]

Define Realization Form
- TDFII

Configure Verification
- Verify limit cycle
- Verification time:3600
- MSP430 16-bit 16 MHz

Verify using a BMC Tool
- ESBMC
- Bound: 10 samples

Result
- SUCESS

SUCCESS

Experimental Objectives

- Use BMC tools to verify digital controllers
- Find potential bugs before the deployment
- Evaluate the proposed methodology, in particular DCVerifier
- Verify overflows, limit cycles, time constraints, and stability
- Verify DFI, DFII, and TDFII implementations

Experiments Setup

- Verification Enviroment
  - Intel Core i7-2600 3.40 GHz processor, 24 GB of RAM, and Ubuntu 11.10 64-bits
  - ESBMC v1.23 with the SMT solver Z3 v4.0
- Hardware Considerations
  - Verifications based on MSP340, 16 MHz clock
  - Sample rate: 100 Hz
  - Wordlength: 16 bits

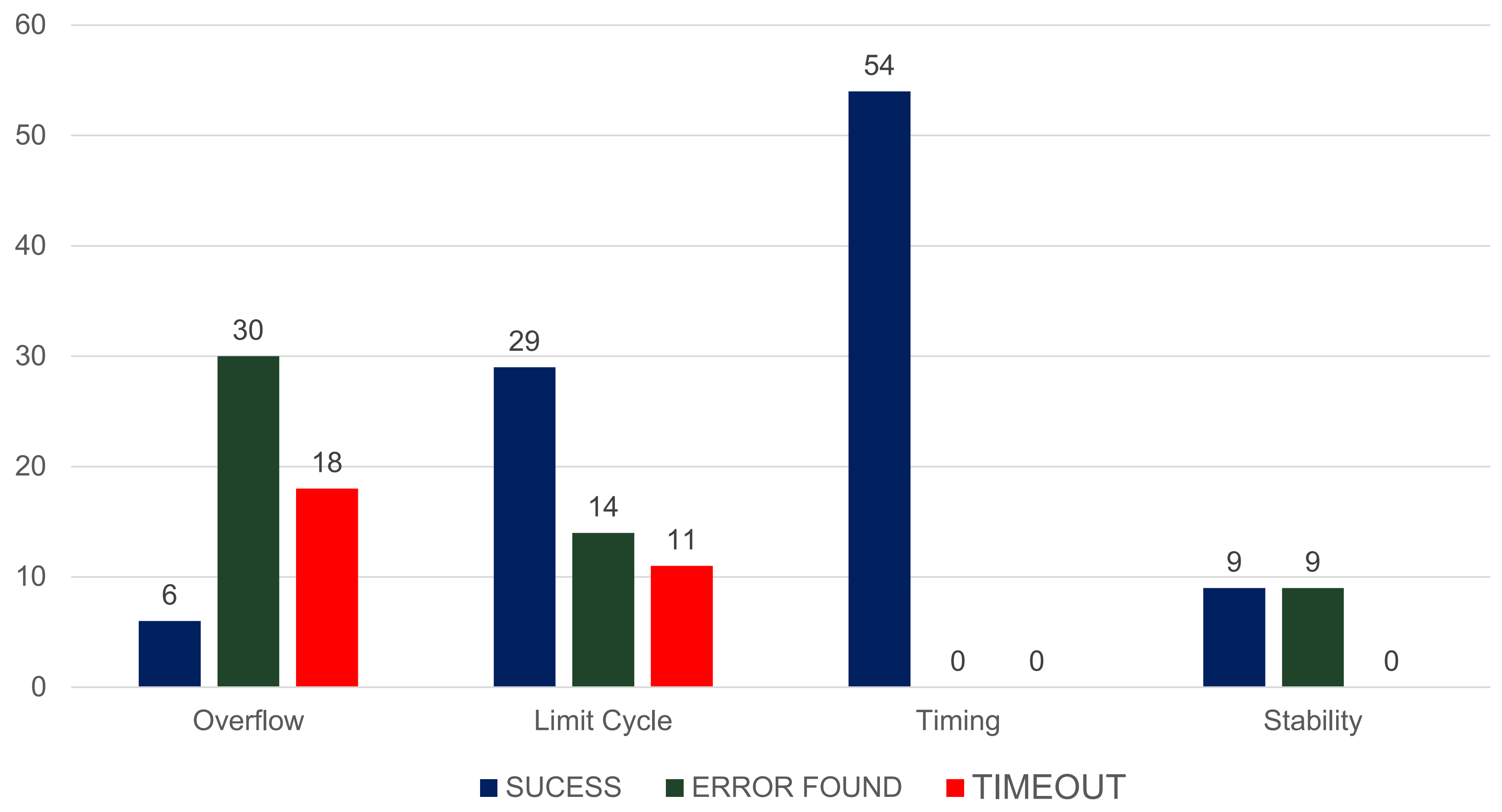
Digital controllers for a Ball and Beam plant

- Quanser Ball and Beam Plant: a track on which the metal ball is free to roll. The Ball and Beam transfer function:

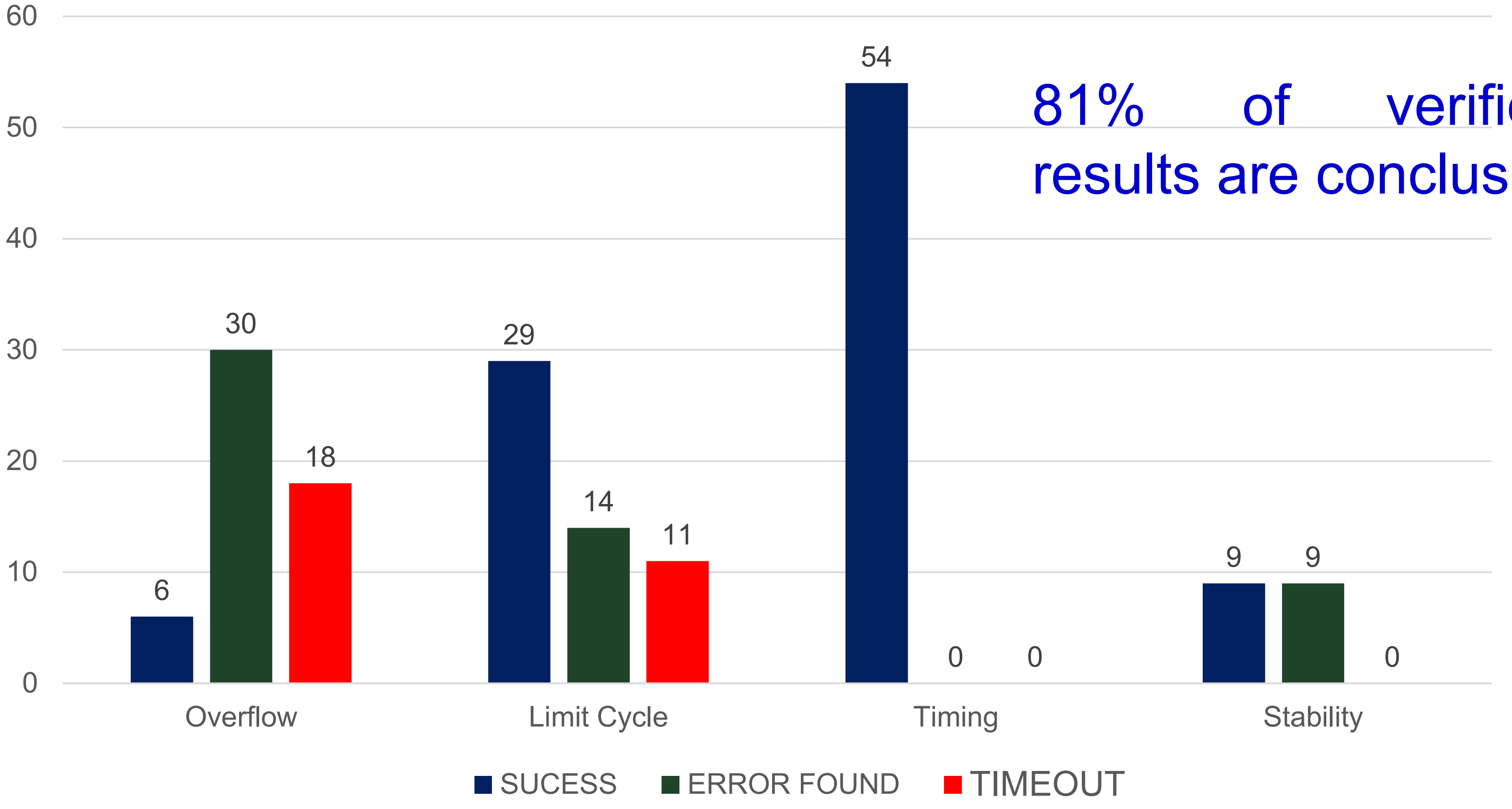
$$G(z) = 1.0067 \times 10^{-8} \frac{(z+9.256)(z+0.9324)(z+0.9389)}{(z-1)^3(z-0.7041)}$$

- 6 different digital controllers were designed, which produced 180 benchmarks
- 18 different numeric representations in fixed-point were designed

Experimental Results



Experimental Results



81% of verifications results are conclusive!

Conclusions

- BMC is a promising alternative for digital controllers verification
- The verifications are conclusive in 81% of the benchmarks
- Neither false positives nor false negatives are reported
- The DCVerifier may reduce the design efforts
  - Since it is automatic and reliable
- Future work
  - Include more properties
  - Include more realization forms
  - Include closed-loop properties verification

Thank you for your attention!

The tool and all benchmarks are avaliable at www.esbmc.org