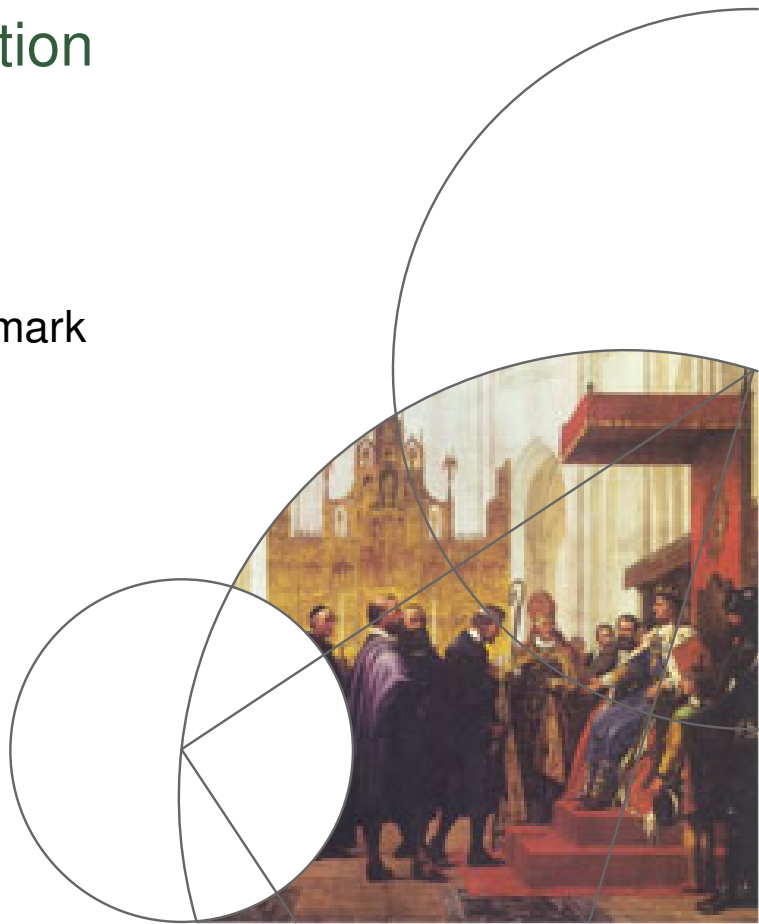
Master's thesis defence

Towards Automatic Program Specification From SME Models

Alberte Thegler
Niels Bohr Institute, University of Copenhagen, Denmark



Overview

Introduction

Motivation

SME was created to bridge the gap between traditional developers and hardware development.

Motivation

SME was created to bridge the gap between traditional developers and hardware development.

Traditional software developers are not trained in same level of testing as hardware developers.

Motivation

SME was created to bridge the gap between traditional developers and hardware development.

Traditional software developers are not trained in same level of testing as hardware developers.

The rising need for automatic testing.

Unfortunate failures

Therac-25 (1980's): Unfortunate key combination and counter overflow caused hardware not to set up properly.

Unfortunate failures

Therac-25 (1980's): Unfortunate key combination and counter overflow caused hardware not to set up properly.

The Patriot Missile Failure (1991): Converting a integer to a floating point number within a 24 bit register.

Unfortunate failures

Therac-25 (1980's): Unfortunate key combination and counter overflow caused hardware not to set up properly.

The Patriot Missile Failure (1991): Converting a integer to a floating point number within a 24 bit register.

Ariane-5 (1996): Converting a 64-bit floating point to signed 16-bit integer.

Unfortunate failures

Therac-25 (1980's): Unfortunate key combination and counter overflow caused hardware not to set up properly.

The Patriot Missile Failure (1991): Converting a integer to a floating point number within a 24 bit register.

Ariane-5 (1996): Converting a 64-bit floating point to signed 16-bit integer.

This is still a problem today.

Why should we verify hardware?

Because, as these examples have shown, the consequences of not verifying can be devastating.

Loss of milions of money.

Loss of human life.

TAPS

A transpiler (Source-to-source compiler) which transpiles SMEIL to CSP_M in order to verify SME models with FDR4.

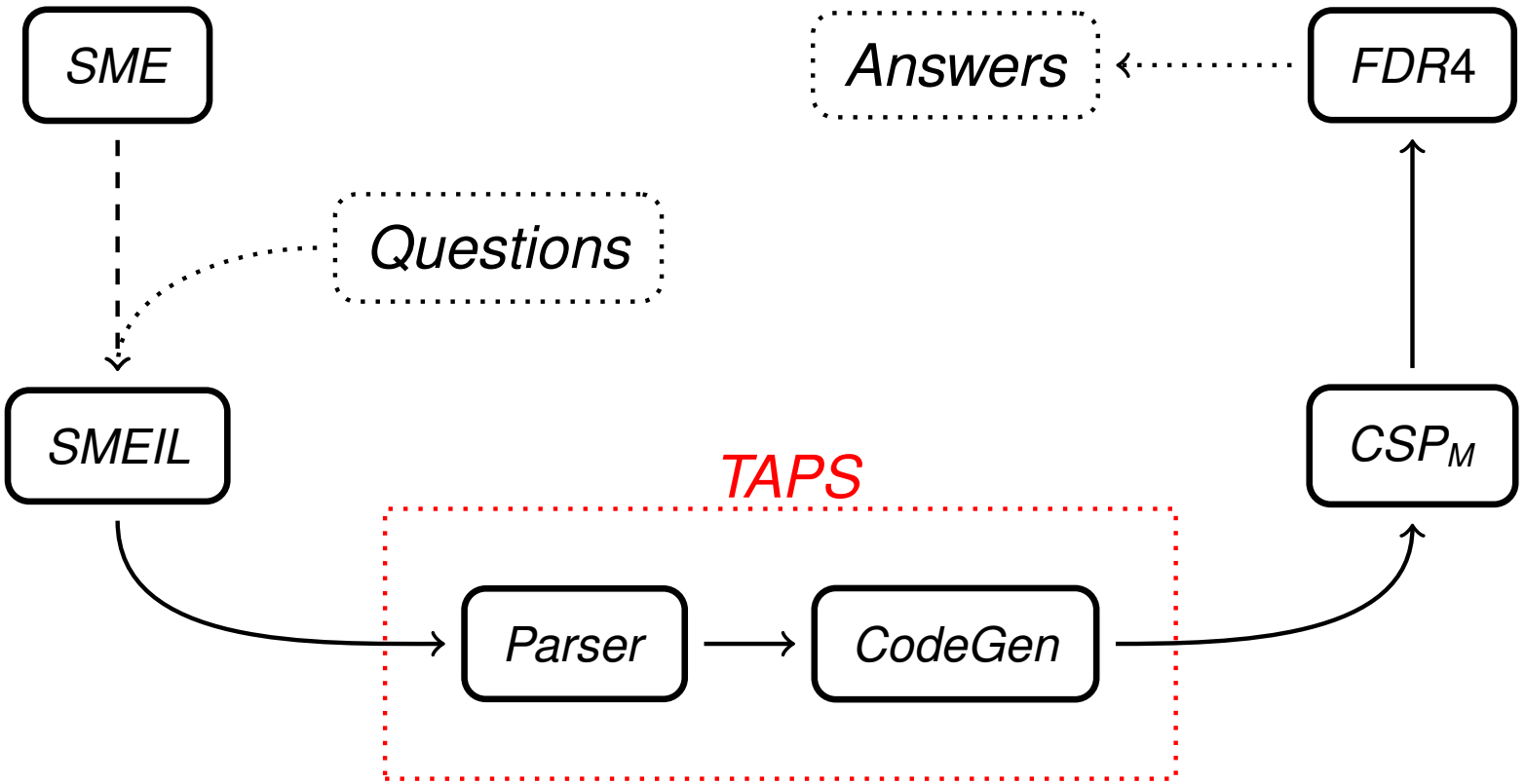


Figure. SME to CSP_M transpiler.

TAPS

Original TAPS

- Formally verify the communication on CSP_M channels
- One clock cycle verification

TAPS

Original TAPS

- Formally verify the communication on CSP_M channels
- One clock cycle verification

Clocked TAPS

- Several clock cycles verification
- simulating a global synchronous clock
- Buffer structure

TAPS

Original TAPS

- Formally verify the communication on CSP_M channels
- One clock cycle verification

Clocked TAPS

- Several clock cycles verification
- simulating a global synchronous clock
- Buffer structure

TAPS removes the need to manually write tests for the hardware model.

TAPS

Original TAPS

- Formally verify the communication on CSP_M channels
- One clock cycle verification

Clocked TAPS

- Several clock cycles verification
- simulating a global synchronous clock
- Buffer structure

TAPS removes the need to manually write tests for the hardware model.

TAPS also provides better coverage than the standard tests.

Refinement Assertions

Refinement assertion

The traces model: Tell us what the process can do

Refinement assertion

The traces model: Tell us what the process can do

The failures-divergences model: If a process can reach itself. We expect our clocked processes to recurse.

Refinement assertion

The traces model: Tell us what the process can do

The failures-divergences model: If a process can reach itself. We expect our clocked processes to recurse.

The failures model: Look at refusal sets for each trace.

Dummy Values

Dummy value

CSP_M code:

```
1  channel sync
2  channel read : {0..15}.Bool
3
4  Id(i, input_channel) =
5      (sync ->
6       input_channel ? x.dummy ->
7       sync ->
8          if (dummy == false) -- initial value
9              then (
10                 i <= 15 & -- upper limit
11                     read ! i.true -> Id(i, input_channel))
12             else (
13                 x <= 15 & -- upper limit
14                     read ! x.true -> Id(i, input_channel))
15     )
16     [] SKIP
17
18
19 read_monitor(c) =
20     (c ? x.dummy ->
21     (0 <= x and x <= 10) & -- observed values + initial value
22         read_monitor(c)
23     ) [] SKIP
```

Linedetector Example

Linedetector in SMEIL

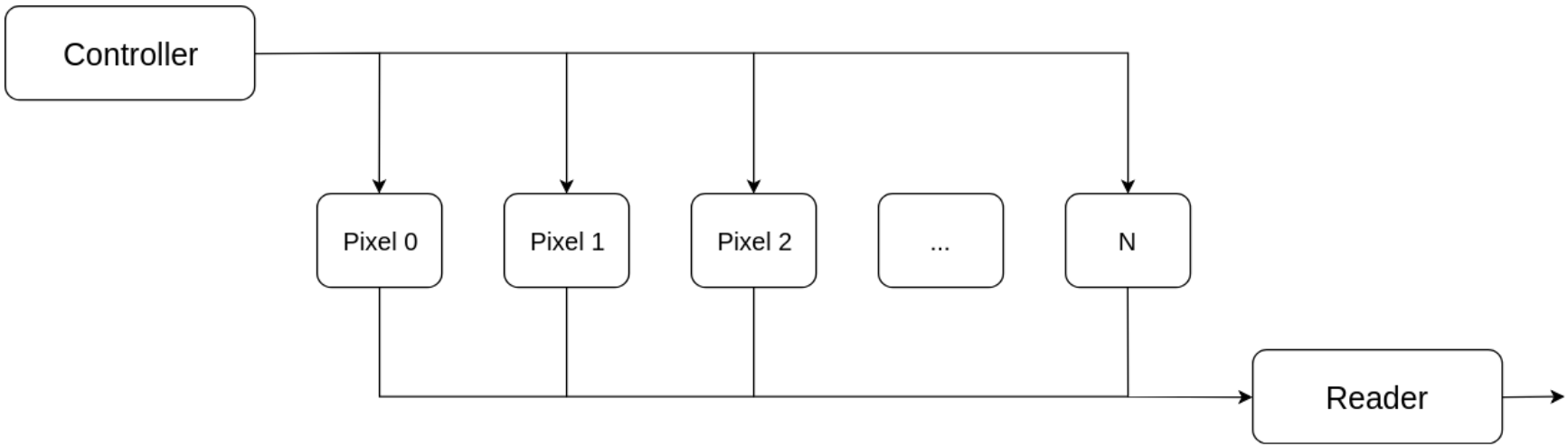


Figure. The linedetector structure.

Linedetector in CSP_M

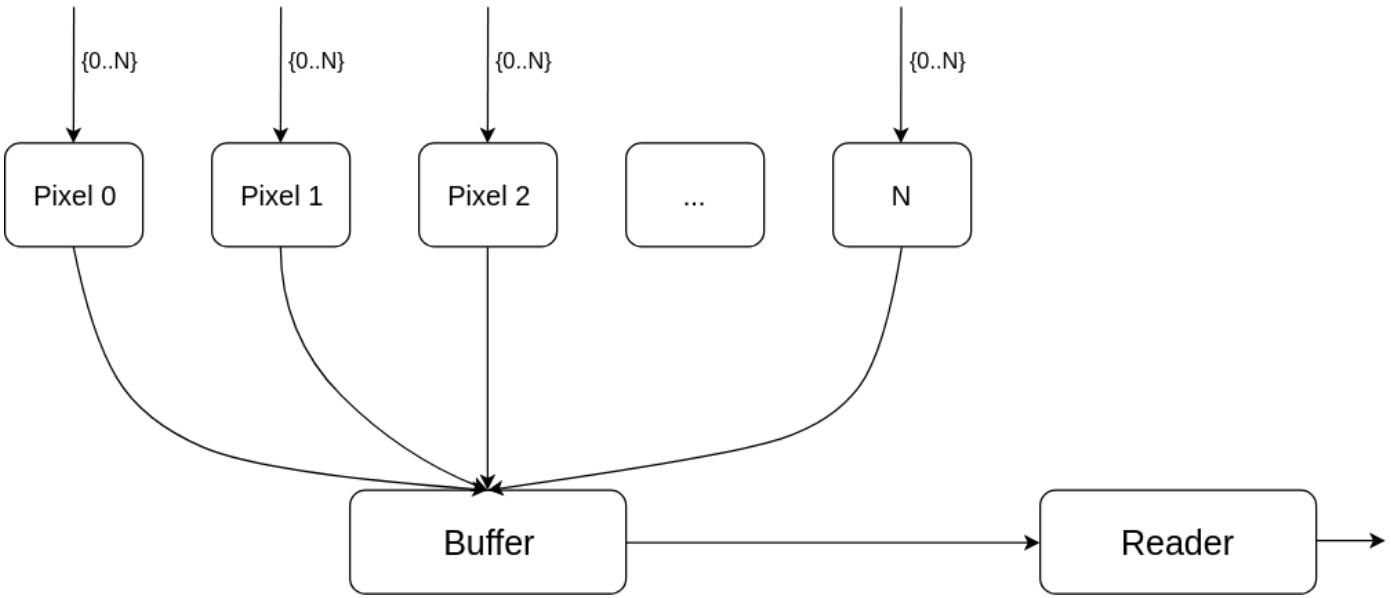


Figure. The linedetector structure with buffer.

Linedetector process code

CSP_M code:

```
1  Pixel0 =
2      (sync ->
3       counter ? x ->
4       sync ->
5       if (x == 0)
6           then sink_r_0 ! 0 -> Pixel0
7           else Pixel0
8      ) [] SKIP
9
10 Pixel1 =
11     (sync ->
12      counter ? x ->
13      sync ->
14      if (x == 0) -- Should be 1 to succeed
15          then sink_r_1 ! 1 -> Pixel1
16          else Pixel1
17     ) [] SKIP
18
19      ⋮
```

Linedetector buffer code

CSP_M code:

```
1  Read = sync -> ((Read_mul(false,0) [] sync -> Read) [] SKIP)
2
3  Read_mul(false, n) = sink_r_0 ? x -> Read_mul(true, x)
4                     [] sink_r_1 ? x -> Read_mul(true, x)
5                     [] sink_r_2 ? x -> Read_mul(true, x)
6                     [] sink_r_3 ? x -> Read_mul(true, x)
7                     [] sink_r_4 ? x -> Read_mul(true, x)
8
9  Read_mul(true, x) = sink_r_0 ? y -> STOP
10                   [] sink_r_1 ? y -> STOP
11                   [] sink_r_2 ? y -> STOP
12                   [] sink_r_3 ? y -> STOP
13                   [] sink_r_4 ? y -> STOP
14                   [] Write(x)
15
16 Writes(x) = sink_w ! x -> (Writes(x) [] Read)
17 Write(x) = sync -> (Writes(x) [] Read) [] SKIP
```

Linedetector verification

Video

Runtime results

Seven segment display clock

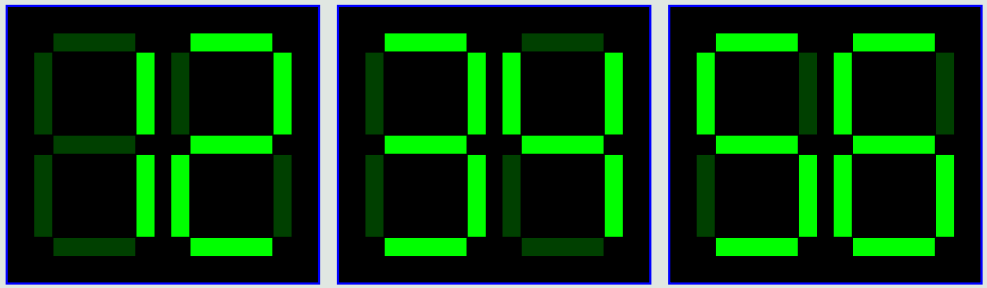


Figure. Digital clock with six seven segment displays, displaying 12:34:56.

Seven segment display clock

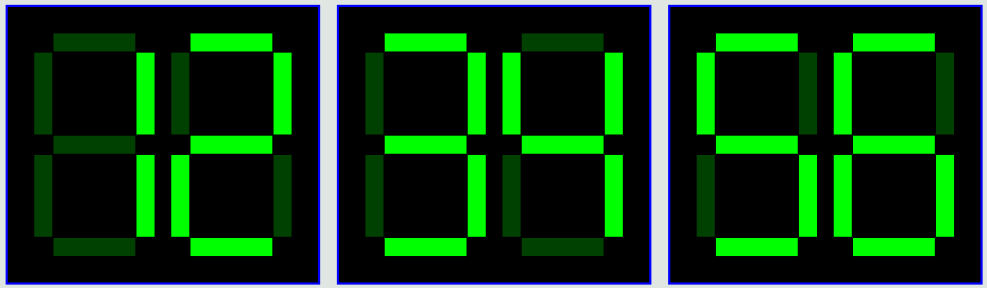


Figure. Digital clock with six seven segment displays, displaying 12:34:56.

Seconds since midnight are calculated into hours, minutes and seconds respectively.

Seven segment display clock

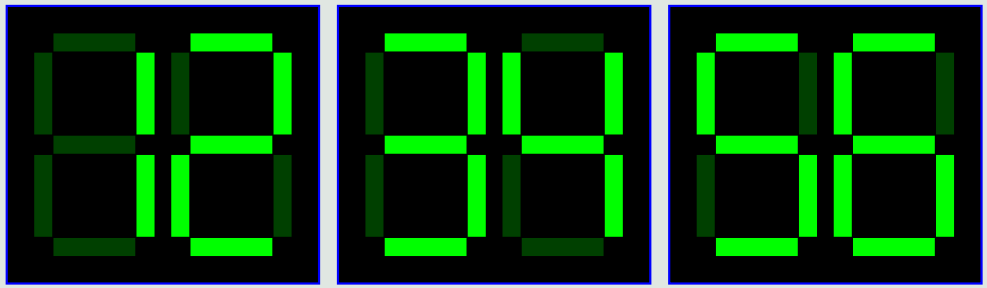


Figure. Digital clock with six seven segment displays, displaying 12:34:56.

Seconds since midnight are calculated into hours, minutes and seconds respectively.

One seven segment display can only display the numbers 0-9.
4 bits can represent 0-15, which is more than needed.

Seven segment display clock

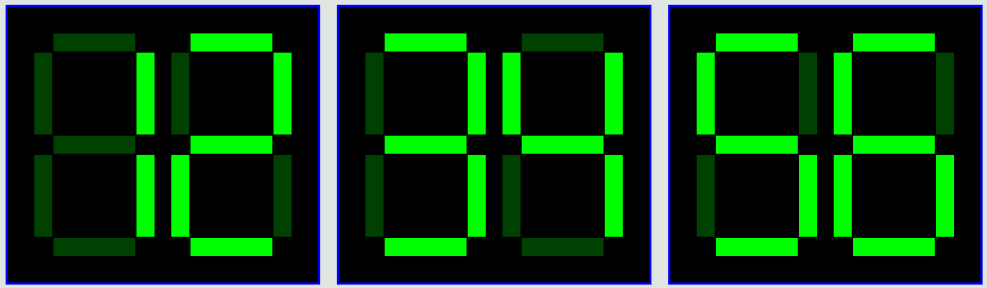


Figure. Digital clock with six seven segment displays, displaying 12:34:56.

Seconds since midnight are calculated into hours, minutes and seconds respectively.

One seven segment display can only display the numbers 0-9.
4 bits can represent 0-15, which is more than needed.

We can verify that the values communicated does not exceed the expected values.

Number of visited states - Clocked and Original

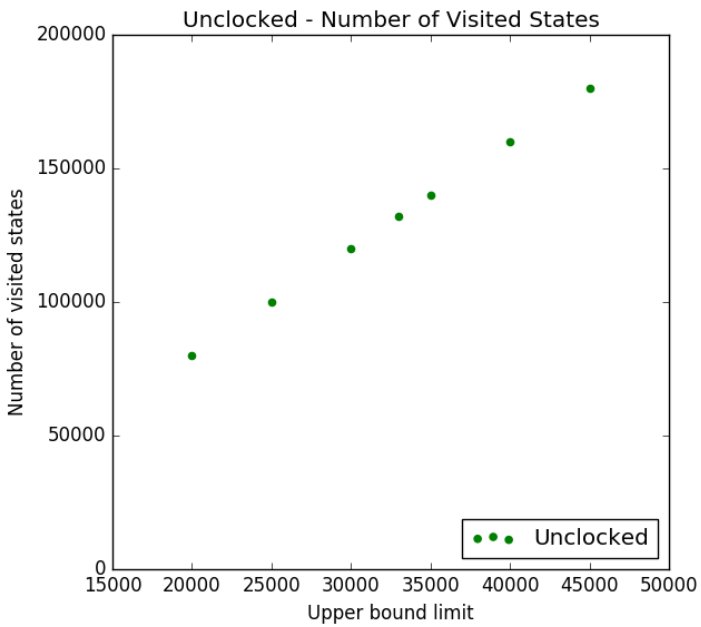


Figure. Unclocked number of visited states

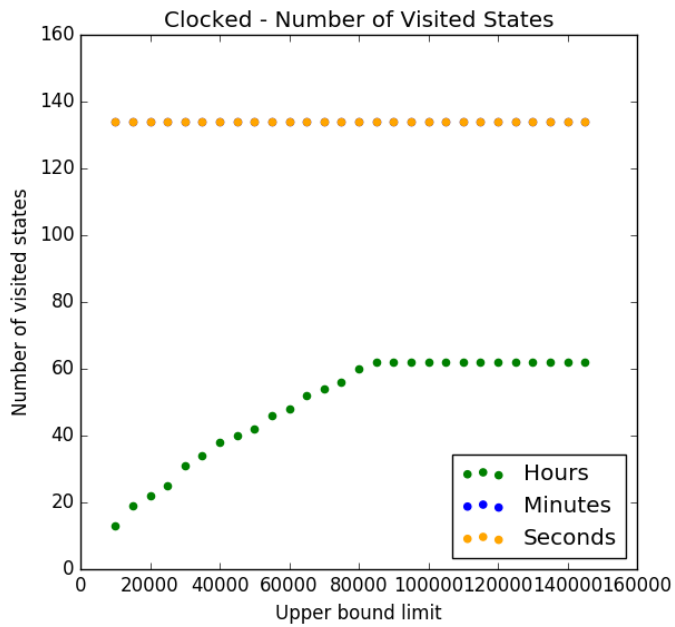


Figure. Clocked number of visited states

Verification time - Clocked and Original

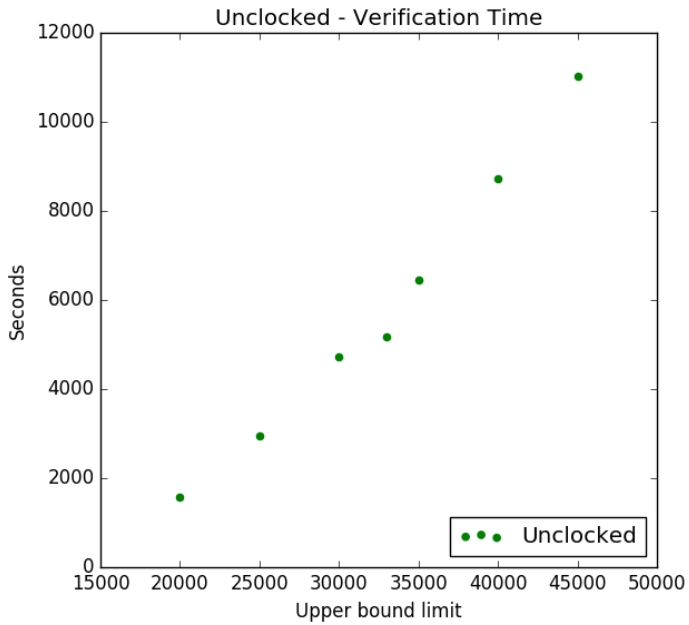


Figure. Unclocked verification time

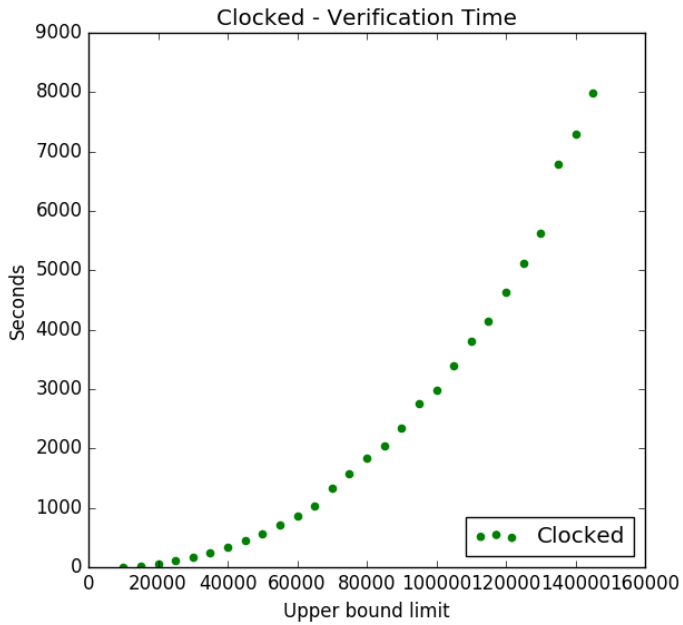


Figure. Clocked verification time

Maximum resident set size - Clocked and Original

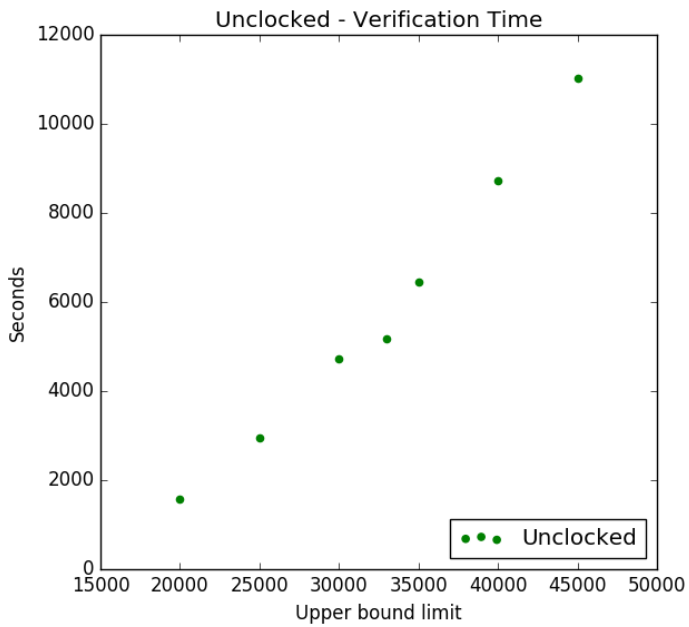


Figure. Unclocked maximum resident set size

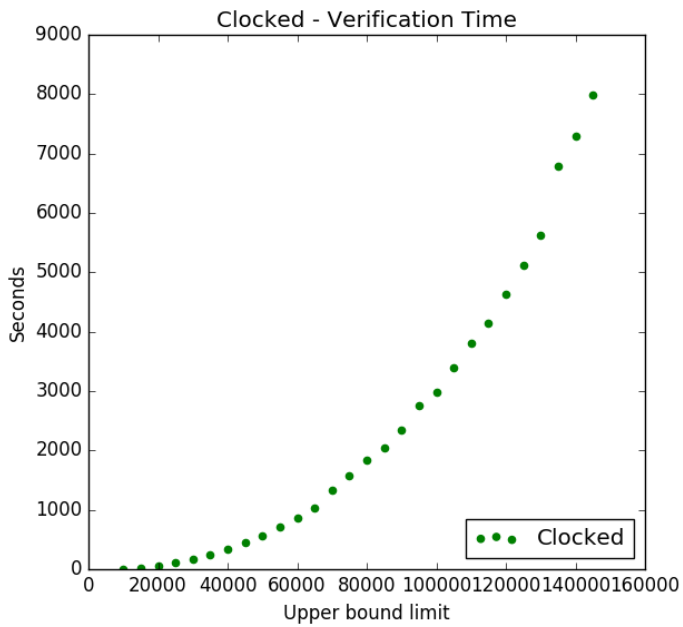


Figure. Clocked maximum resident set size

Increase in processes - Combined

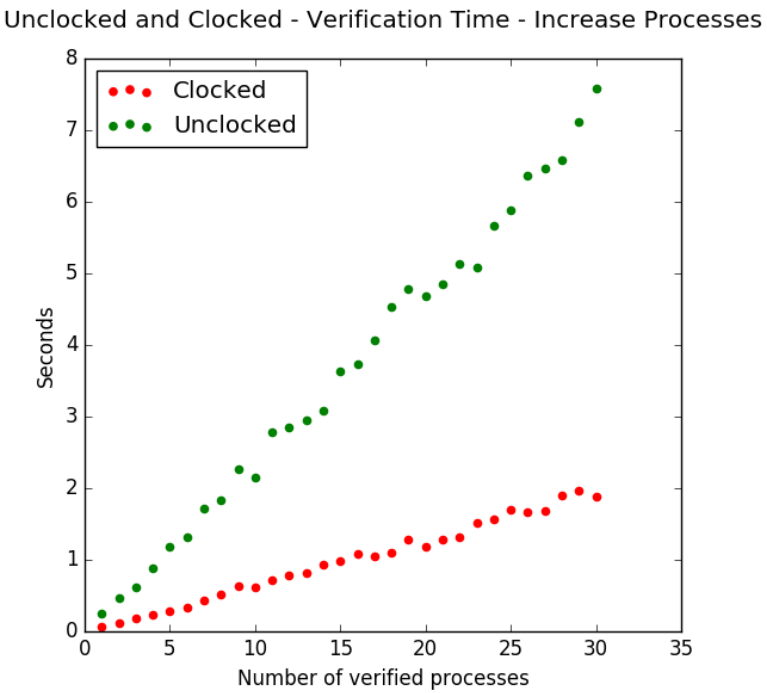


Figure. Combined verification time

Increase in processes - Combined

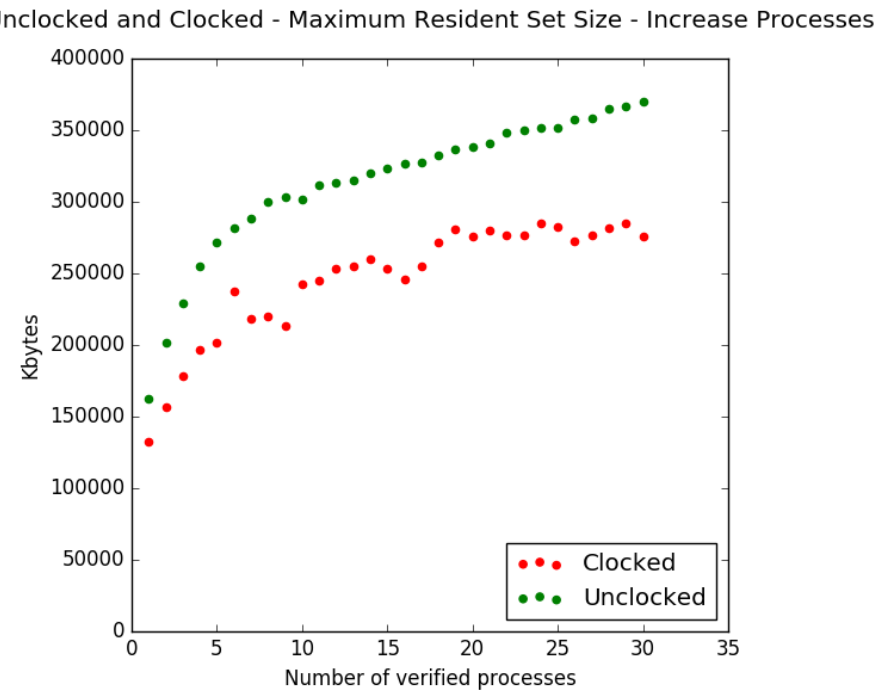


Figure. Combined maximum resident set size

How can we use this?

The original version does not seem to be feasible to use, but the clocked version show great promise.

How can we use this?

The original version does not seem to be feasible to use, but the clocked version show great promise.

More experimentation is needed.

Future work

Future work

Advanced assertions.

Future work

Advanced assertions.

Extend for co-simulation.

Future work

Advanced assertions.

Extend for co-simulation.

Exchange verified elements with equivalent smaller processes might reduce verification time.

Thank you!