Master’s thesis defence

Towards Automatic Program Specification From SME Models

Alberte Thegler
Niels Bohr Institute, University of Copenhagen, Denmark

Overview

1. Introduction
2. Refinement Assertions
3. Dummy Values
4. Linedetector Example
5. Runtime results
6. Future work

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 caused hardware not to set up properly.

Unfortunate failures

Therac-25 (1980’s): Unfortunate key combination caused hardware not to set up properly.

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

Unfortunate failures

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

The Patriot Missile Failure (1991): Converting a integer to a real 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 caused hardware not to set up properly.

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

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

Still a problem today, but hardware problems might be kept out of the press.

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.

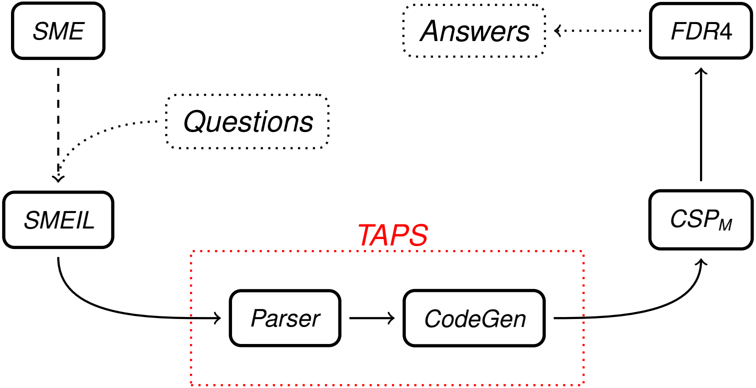

Figure. SME to CSP$_M$ transpiler.

TAPS

Original TAPS

- Formally verify the widths of CSP$_M$ channels
- One clock cycle verification

TAPS

Original TAPS

- Formally verify the widths of 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 widths of 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 widths of 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.

Further introduction to TAPS and the clocked structure?

??

Refinement Assertions

Refinement assertion

The traces model:

$\mathit{traces}(\mathit{STOP}) = \{\langle\rangle\}$

$\mathit{traces}(\checkmark \rightarrow \mathit{STOP}) = \{\langle\rangle, \langle\checkmark\rangle\}$

$\{\langle\rangle\} \in \{\langle\rangle, \langle\checkmark\rangle\}$

Refinement assertion

The traces model:
$\textit{traces}(\textit{STOP}) = \{\langle\rangle\}$
$\textit{traces}(\checkmark \rightarrow \textit{STOP}) = \{\langle\rangle, \langle\checkmark\rangle\}$
$\{\langle\rangle\} \in \{\langle\rangle, \langle\checkmark\rangle\}$

The failures model:
$\textit{failures}(\textit{STOP}) = ?$
$\textit{failures}(\checkmark \rightarrow \textit{STOP}) = ?$
(one is not in the other)

Refinement assertion

The traces model:
$\mathit{traces}(\mathit{STOP}) = \{\langle\rangle\}$
$\mathit{traces}(\checkmark \rightarrow \mathit{STOP}) = \{\langle\rangle, \langle\checkmark\rangle\}$
$\{\langle\rangle\} \in \{\langle\rangle, \langle\checkmark\rangle\}$

The failures model:
$\mathit{failures}(\mathit{STOP}) =?$
$\mathit{failures}(\checkmark \rightarrow \mathit{STOP}) =?$
(one is not in the other)

The failures-divergences model: We expect our processes to diverge

Dummy Values

Dummy value

(code)

Linedetector Example

Linedetector in SMEIL

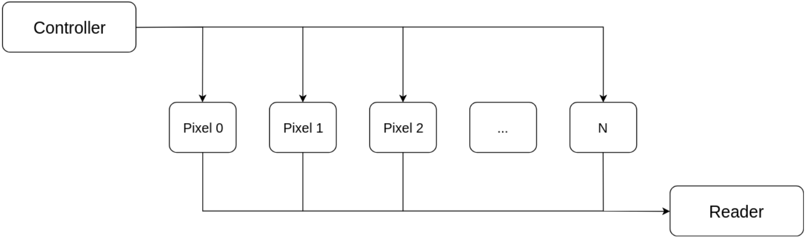

Figure. The linedetector structure.

Linedetector in CSP$_M$

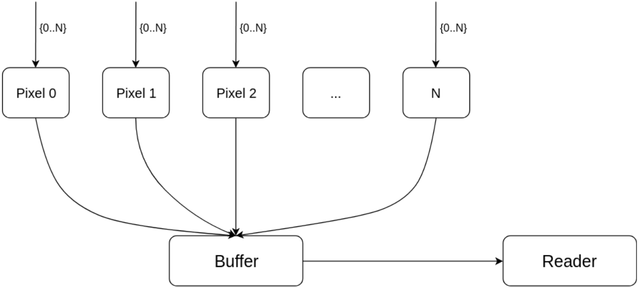

Figure. The linedetector structure with buffer.

Linedetector process code

CSP$_M$ code:

```
Pixel0 =
    (sync ->
     counter ? x ->
     sync ->
     if (x == 0)
         then sink_r_0 ! 0 -> Pixel0
         else Pixel0
    ) [] SKIP

Pixel1 =
    (sync ->
     counter ? x ->
     sync ->
     if (x == 0) -- Should be 1 to succeed
         then sink_r_1 ! 1 -> Pixel1
         else Pixel1
    ) [] SKIP

     ⋮
```

Linedetector buffer code

CSP$_M$ code:

```
Read = sync -> ((Read_mul(false,0) [] sync -> Read) [] SKIP)

Read_mul(false, n) = sink_r_0 ? x0 -> Read_mul(true, x0)
                   [] sink_r_1 ? x1 -> Read_mul(true, x1)
                   [] sink_r_2 ? x2 -> Read_mul(true, x2)
                   [] sink_r_3 ? x3 -> Read_mul(true, x3)
                   [] sink_r_4 ? x4 -> Read_mul(true, x4)

Read_mul(true, x) = sink_r_0 ? y0 -> STOP
                  [] sink_r_1 ? y1 -> STOP
                  [] sink_r_2 ? y2 -> STOP
                  [] sink_r_3 ? y3 -> STOP
                  [] sink_r_4 ? y4 -> STOP
                  [] Write(x)

Writes(x) = sink_w ! x -> (Writes(x) [] Read)
Write(x) = sync -> (Writes(x) [] Read) [] SKIP
```

Linedetector verification

Video

Runtime results

Seven segment display clock

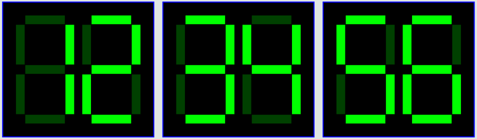

Figure. Digital clock with six seven segment displays, displaying 12:34:56.

Seven segment display clock

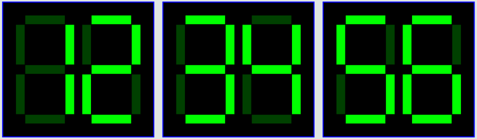

Figure. Digital clock with six seven segment displays, displaying 12:34:56.

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

Seven segment display clock

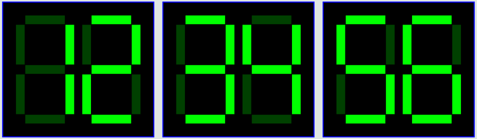

Figure. Digital clock with six seven segment displays, displaying 12:34:56.

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

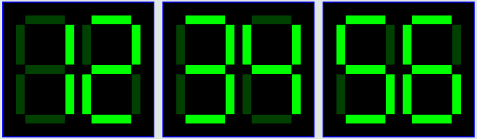

Figure. Digital clock with six seven segment displays, displaying 12:34:56.

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.

Original Seven Segment Display Example Runtime

Clocked Seven Segment Display Example Runtime

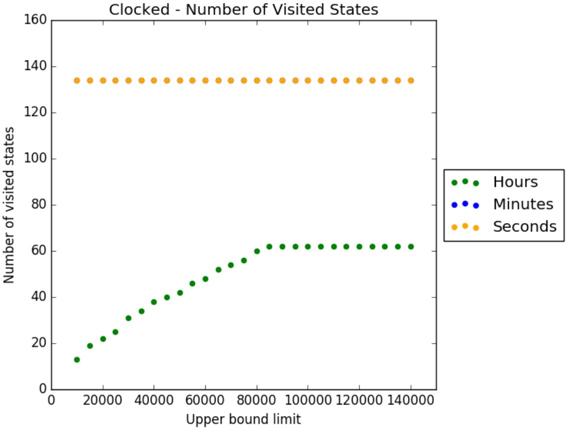

Figure. Clocked seven segment display

Clocked Seven Segment Display Example Runtime

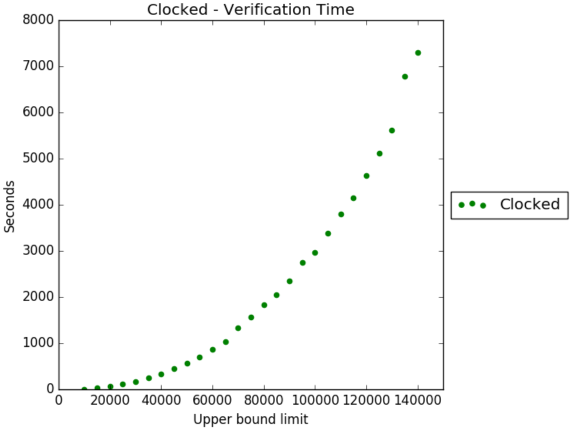

Figure. Clocked seven segment display

Clocked Seven Segment Display Example Runtime

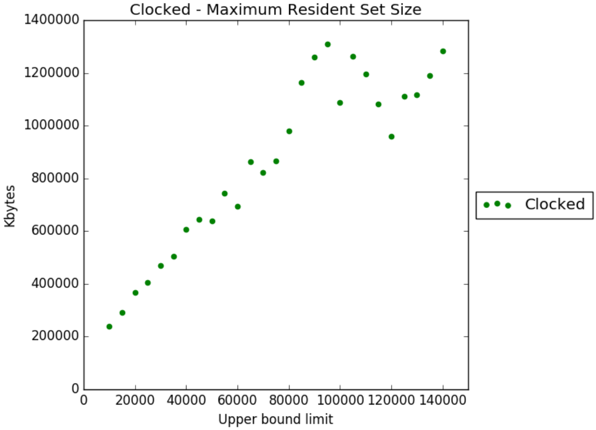

Figure. Clocked seven segment display

Unclocked vs. clocked

Original Seven Segment Display Example benchmark

Clocked Seven Segment Display Example benchmark

Unclocked vs. clocked

How can we use this?

(Update when I have all data) Original vs. clocked - big difference

How can we use this?

(Update when I have all data) Original vs. clocked - big difference

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

How can we use this?

(Update when I have all data) Original vs. clocked - big difference

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

Compositioning with CSP processes might reduce verification time

Thank you!