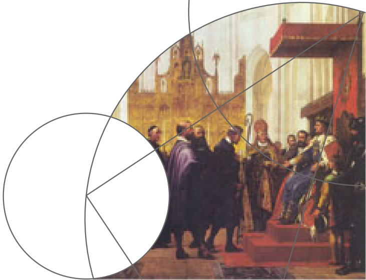
Faculty of Science

Towards Automatic Program Specification Using SME Models

Communicating Process Architectures 2018 – Technische Universität Dresden

Alberte Thegler
Niels Bohr Institute, University of Copenhagen, Denmark



21 August 2018

Why should we verify hardware?

Ariane-5

4th June 1996

Why should we verify hardware?

Ariane-5

4th June 1996

Total failure on launch

Why should we verify hardware?

Ariane-5

4th June 1996

Total failure on launch

Converting a 64-bit floating point number to signed 16-bit integer.

Why should we verify hardware?

Ariane-5

4th June 1996

Total failure on launch

Converting a 64-bit floating point number to signed 16-bit integer.

Overflow caused the self-destruct mechanism in both primary and backup computer

Why should we verify hardware?

Ariane-5

4th June 1996

Total failure on launch

Converting a 64-bit floating point number to signed 16-bit integer.

Overflow caused the self-destruct mechanism in both primary and backup computer

No people where harmed

Why should we verify hardware?

The Patriot Missile Failure

25th February 1991 in the Persian Gulf war

Why should we verify hardware?

The Patriot Missile Failure

25th February 1991 in the Persian Gulf war

A Patriot missile failed to intercept an incomming ”Scud” which struck a U.S Army barracks, killing 28 soldiers.

Why should we verify hardware?

The Patriot Missile Failure

25th February 1991 in the Persian Gulf war

A Patriot missile failed to intercept an incomming ”Scud” which struck a U.S Army barracks, killing 28 soldiers.

A bug in the system’s weapons control computer caused an inaccurate tracking calculation. Conversion of time since last boot from an integer to a real number was performed using a 24 bit register.

Why should we verify hardware?

The Patriot Missile Failure

25th February 1991 in the Persian Gulf war

A Patriot missile failed to intercept an incomming ”Scud” which struck a U.S Army barracks, killing 28 soldiers.

A bug in the system’s weapons control computer caused an inaccurate tracking calculation. Conversion of time since last boot from an integer to a real number was performed using a 24 bit register.

Inaccurate results == missile misses target

What have we done?

A transpiler which transpiles SMEIL code to CSP_M in order to verify SME models with FDR4

SME

The SME model builds on the CSP algebra what more to add?

SMEIL

You have just been introduced to SMEIL in the previous presentation

SMEIL

You have just been introduced to SMEIL in the previous presentation

We transpile from SMEIL to CSP$_M$
And then verify it in FDR4

SMEIL

You have just been introduced to SMEIL in the previous presentation

We transpile from SMEIL to CSP$_M$
And then verify it in FDR4

Currently only works with pure SMEIL programs

SMEIL

You have just been introduced to SMEIL in the previous presentation

We transpile from SMEIL to CSP_M
And then verify it in FDR4

Currently only works with pure SMEIL programs

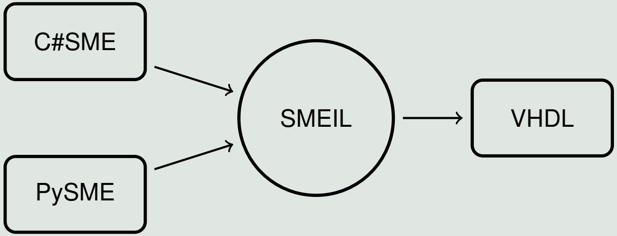


Figure. SMEIL transpiler structure.

Seven segment display clock

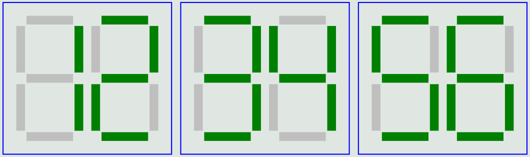


Figure. Digital clock with six seven segment displays, displaying 12:34:56.

Seven segment display clock

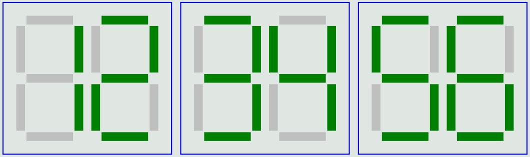


Figure. Digital clock with six seven segment displays, displaying 12:34:56.

Seconds since midnight

Seven segment display clock

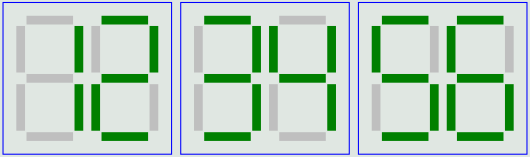


Figure. Digital clock with six seven segment displays, displaying 12:34:56.

Seconds since midnight

Arithmetics calculate hours, seconds and minutes respectively

Seven segment display clock

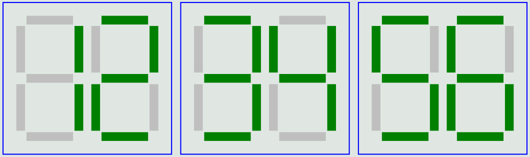


Figure. Digital clock with six seven segment displays, displaying 12:34:56.

Seconds since midnight

Arithmetics calculate hours, seconds and minutes respectively

Two seven segment displays pr. `time` process

Simple example

What are we verifying

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

Simple example

What are we verifying

One seven segment example 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 to the seven segment displays does not exceed the expected values.

Simple example

What are we verifying

One seven segment example 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 to the seven segment displays does not exceed the expected values.

In general, we verify the values comnmunicated on CSP$_M$ channels

Simple example

What are we verifying

One seven segment example 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 to the seven segment displays does not exceed the expected values.

In general, we verify the values comnmunicated on CSP$_M$ channels

In this case we can restrict the assertions further.
Hours will never be more than 24, etc.

Simple example

Seven Segments SMEIL Structure

SMEIL code:

```
proc seconds (in seconds_in)
    bus seconds_out {first_digit: u3 range 0 to 5;
                     second_digit: u4 range 0 to 9;};
    var seconds: u6 range 1 to 59;
    var seconds_first_temp: u3 range 0 to 5;
    var seconds_second_temp: u4 range 0 to 9;
{
    seconds = seconds_in.val % 60;
    seconds_first_temp = seconds / 10;
    seconds_second_temp = seconds % 10;
    seconds_out.first_digit = seconds_first_temp;
    seconds_out.second_digit = seconds_second_temp;
}
```

Simple example

Seven Segments SMEIL Structure

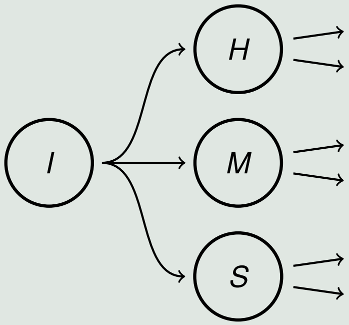


Figure. SMEIL network for a seven segment display clock. Each SMEIL process is represented by a cicle with a letter corresponding to the processes Input, Hours, Minutes and Seconds respectively.

SMEIL bus to CSP$_M$ channel

SMEIL code:

```
channel seconds_out_first_digit : {0..7}
channel seconds_out_second_digit : {0..15}

    ⋮

Seconds(seconds_in) =
let
    seconds = seconds_in % 60
    seconds_first_temp = seconds / 10
    seconds_second_temp = seconds % 10
within
    seconds_out_first_digit ! seconds_first_temp ->
    seconds_out_second_digit ! seconds_second_temp ->
    SKIP
```

CSP$_M$ process structure

Code example

Monitor process

SMEIL code:

```
Seconds_out_first_digit_monitor(c) =
    c ? x -> if 0 <= x and x <= 5 then SKIP else STOP
Seconds_out_second_digit_monitor(c) =
    c ? x -> if 0 <= x and x <= 9 then SKIP else STOP
```

Example continued

CSP_M code

CSP_M code? Do we even need this?

Results - time to verify in FDR4?

The seven segment example have been run on a Intel(R) Xeon(R) CPU E5-2698 v4 @ 2.20GHz.

The example were run x times and the average was measured. (If I have time)

Conclusion

With this system we can transpile hardware models to CSP$_M$

Conclusion

With this system we can transpile hardware models to CSP_M

and verify values on the CSP_M channels

Conclusion

With this system we can transpile hardware models to CSP$_M$

and verify values on the CSP$_M$ channels

and thereby verify the original hardware model

Future work

Rest of SMEIL grammar?

+ more?

Questions?

Thank you!

Thank you so much for your time.
Feel free to ask anything.