A Timing Model for Synchronous Language Implementations in Simulink

Timothy Bourke and Arcot Sowmya
School of Computer Science and Engineering
University of New South Wales, Sydney
and
National ICT Australia
tbourke@cse.unsw.edu.au
Outline

⇒ Simulink and Stateflow

An Argos block

Timing Model

Embedding within Simulink

Concluding remarks
Simulink and Stateflow

- Popular tools
- Practical focus
- Several shortcomings

Simulation $\leftrightarrow$ Model-driven Development
Reasoning about Stateflow designs is complicated:

1. intricate ordering rules
2. queued event processing
3. stacking of communications
4. implicit assumption of synchrony

Synchronous languages have better underlying models (assumption)
An Argos [Mar91, MR01] block: syncblock [BS05]

- Our first attempt at combining synchronous languages and Simulink.
- Simulate with Argos controllers.

Aside: [CCM⁺03, SSC⁺04]

- integrate rather than extract
- simulate sync. programs
syncblock: simulating embedded controllers

- **Original Prototype**: perfect synchrony
  - Block outputs appear simultaneously with inputs.
  - i.e. in the same Simulink step.

- *But*, Simulink normally models timing detail.

**consider: dedicated embedded controllers**

*aim*: provide simulation runs with low-level timing detail.
**Revised approach**: simulate implementation delays.

- Internally: synchronous semantics.
- Externally: delay between inputs and outputs.

- Necessary to **latch** inputs and outputs, and to **schedule** reactions.
  - Effectively modelling part of the platform (if abstractly).
Idealised parameters

- event-driven or sample-driven: mode
- Delay between input and output: $\delta_{out}$
- Minimum pause between reactions: $\tau$
- Program + Limitations $\rightarrow$ Simulation (block)
  $\rightarrow$ Implementation (model)

aside: TAXYS [STY03]
Outline

✓ Simulink and Stateflow
✓ An Argos block
⇒ Timing Model

Embedding within Simulink

Concluding remarks
Transformation to Timed Automata

fix: \[ A_B = \langle S, s_0, I, O, T \rangle \quad \tau \in \mathbb{Q}_0^+ \]
\[ \text{trigger} \in \{ \text{sample, event} \} \quad \delta_{out} \in \mathbb{Q}_0^+ \]

requiring: \[ \delta_{out} \leq \tau \]
\[ \text{trigger} = \text{event} \lor \tau > 0 \]

then define: \[ A_{\tau,\delta_{out}}^{\text{trigger}} = \langle \Sigma, L, L_0, C, E \rangle: \quad \text{[AD94]} \]

- \[ \Sigma = I \cup O \cup \{ \text{react} \} \]
- \[ L = (S \cup \{ \text{startup} \}) \times P(I) \times P(O) \times \mathbb{B} \]
- \[ L_0 = \{ (\text{startup}, \emptyset, \emptyset, \text{ff}) \} \]
- \[ C = \{ x \} \]
- \[ E \] is the smallest set defined by the conjunction of 9 transition rules.
(almost) ABRO [Ber00]: in Argos
(almost) ABRO [Ber00]: in Argos

```
1 loop
2 [ 3
3 4
4 ||
5 6
6 ];
7 emit o
8 each r
```

```
ABRO(a, b, r)(o)
```

```
l_1
a/l_1

b/l_2
```

```
\( l_1 \cdot l_2 \cdot \neg r/o \)
```
(almost) ABRO: Labelled Transition System
(almost) ABRO: Timed Transition System
(almost) ABRO: Timed Transition System
(almost) ABRO: Timed Transition System

trigger = sample

\[ x > 0 \]
(almost) ABRO: Timed Transition System

Trigger = sample
(almost) ABRO: Timed Transition System

trigger = sample
(almost) ABRO: Timed Transition System

trigger = sample
Event-driven triggering

\[ trigger = event \]

\[ react, x = \tau, x := 0 \]

\[ react, x = 0 \]

\[ x \leq \tau \]

\[ x = 0 \]

\[ x > \tau, x := 0 \]

- Input events during a reaction must wait until \( x = \tau \)
- Otherwise, they trigger a reaction *urgently* [BST97]
Outline

√ Simulink and Stateflow
√ An Argos block
√ Timing Model
⇒ Embedding within Simulink

Concluding remarks
Embedding within Simulink

One block or many?
Embedding within Simulink

**Adopt a semantics for Simulink**

- Simulation Engine
- Intent of models

**Translate models**

*e.g. to Lustre*

**Interactions of block**

*mix conceptual and low-level operations*
Mathworks Bang-bang temperature controller

1 sec Timer
1 sec Timer

Argos bang-bang controller

LED {OFF=0, RED=1, GREEN=2}

BOILER CMD {OFF=0, ON=1}

TEMP (deg C)

actual temp
digital temp

Boiler
Plant model

1 sec Timer
1 sec Timer

1 sec Timer
1 sec Timer

LED

SEC

reference

2

COLD

syncblock

<

1
temp

syncblock

LED

1

2

boiler

reference

1

SEC

temp

COLD

[5 SEC]

/ LED_RED

[5 SEC]

/ LED_RED

[40 SEC]

/ onOk

COLD.onOk / BOILER.ON

~COLD

SEC

SEC / LED_GREEN

~SEC / LED_GREEN

[20 SEC]

/ BOILER.ON

1 sec Timer
1 sec Timer

1 sec Timer
1 sec Timer

1 sec Timer
1 sec Timer

1 sec Timer
1 sec Timer

1 sec Timer
1 sec Timer

1 sec Timer
1 sec Timer

1 sec Timer
1 sec Timer

1 sec Timer
1 sec Timer

1 sec Timer
1 sec Timer

1 sec Timer
1 sec Timer

1 sec Timer
1 sec Timer

1 sec Timer
1 sec Timer

1 sec Timer
1 sec Timer

1 sec Timer
1 sec Timer

1 sec Timer
1 sec Timer

1 sec Timer
1 sec Timer

1 sec Timer
1 sec Timer
Simulink simulation engine: initialization

1. Flatten model
1. Flatten model
1. Flatten model

2. Order by signal dependencies.
1. Flatten model
2. Order by signal dependencies.
3. Start at $t = 0$.
4. 
5. Visit each block—maybe several times.
6. Increase $t$—depends on solver.
7. repeat from step 5
Behaviour of syncblock

\[
y = f_o(t, x, u) \quad \text{outputs}
\]
\[
x'_d = f_u(t, x, u) \quad \text{update}
\]

\(x_c\) previous clock value
\(x_{tp}\) previous sample time

- Two predicates: \textit{react} and \textit{emit}.

- Instants of interest:
  - \textbf{sample-driven}:
    \[
    \delta_{out} = \tau \quad \begin{array}{|c|c|}
    \hline
    \text{otherwise} & [\tau, 0] \text{ and } [\tau, \delta_{out}] \\
    \hline
    \end{array}
    \]
  - \textbf{event-driven}:
    \[
    \tau = 0 \quad \text{inherited}
    \]
    \[
    \text{otherwise} \quad \text{zero-crossings}
    \]
Effect of parameters

Bang-bang Controller: Stateflow

Temperature
LED
BOILER

simulation time

out = 0

0 100 200 300 400 500 600 700 800
Effect of parameters

Bang-bang Controller: Stateflow

Temperature
LED
BOILER

$\delta_{out} = 0.3$
Summary

✓ Simulink and Stateflow
✓ An Argos block
✓ Timing Model
✓ Embedding within Simulink

⇒ Concluding remarks
  ● Working prototype uses Argos.
  ● Timed automata framework clarifies implementation.
  ● Looking for case-studies to evaluate utility.
Concluding remarks
References


