Updates on SCCharts

Christian Motika • Steven Smyth

SYNCHRON 2015
04. DEC 2015, Kiel
Reactive System

- Safety-critical systems
- State based reactions
- Concurrency

$\Rightarrow$ Synchronous Language

**SCCharts** = SyncCharts Syntax + Sequential Constructive Semantics
Recall SCCharts

Core-SCCharts
Small set of simple features ease down stream compilation

Extended-SCCharts
Rich set of advanced features ease modeling

Christian Motika, Steven Smyth
SYNCHRON '15
SCCharts = SyncCharts Syntax  + Sequential Constructive Semantics

“Look, it’s flying!”
“Yes, I did it with a magic spell.”
ALDO Example

- Interface
- Local Variables
- Signals

- Concurrency
- Instantaneous Communication
- Preemption
Modeling ALDO

SCCharts = SyncCharts Syntax + Sequential Constructive Semantics

Textual View and Editing

Graphical View

[VL/HCC’13]

Abstract Model

Christian Motika, Steven Smyth

SYNCHRON ’15
Modeling ALDO Demo
SCCharts
KIELER Compiler
Usage & Tests
Applications

KIELER Compiler

- Reliable Compiler + Reliable Models + Praticability [SYNCHRON’14]
- **Single-Pass Language-Driven Incremental Compilation (SLIC)** [ISOLA 14]
  - Interactive Model-Transformation-Based Compiler
  - Intermediate Results: White-Box Compiler
KIELER Compiler (2)

Textual Modeling → Modeled Diagram → (Intermediate) Transformed Diagram / Code

Select Transformation

Single-Pass Language-Driven Incremental Compilation (SLIC)

Original Model → Intermediate Model → Fully transformed Model

Christian Motika, Steven Smyth

SYNCHRON '15
### SLIC Order

<table>
<thead>
<tr>
<th>produces</th>
<th>Weak Suspend</th>
<th>Deferred</th>
<th>History</th>
<th>Static</th>
<th>Valued Signal</th>
<th>Pure Signal</th>
<th>Suspender</th>
<th>Pre</th>
<th>Count Delay</th>
<th>During</th>
<th>Complex Final</th>
<th>Abort</th>
<th>Count</th>
<th>Exit</th>
<th>Initialization</th>
<th>Entry</th>
<th>Connector</th>
</tr>
</thead>
<tbody>
<tr>
<td>Weak Suspend</td>
<td>↑</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Deferred</td>
<td>↑</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>History</td>
<td>↑</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Static</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Valued Signal</td>
<td>↑</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Pure Signal</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Suspender</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Pre</td>
<td>&lt;↓</td>
<td>&lt;↓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Count Delay</td>
<td></td>
<td></td>
<td>&lt;↓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>During</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Complex Final</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Abort</td>
<td>&lt;↓</td>
<td>&lt;↓</td>
<td>&lt;↓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Const</td>
<td></td>
<td></td>
<td>&lt;↓</td>
<td>&lt;↓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Exit</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Initialization</td>
<td>&lt;↓</td>
<td>&lt;↓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Entry</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Connector</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Christian Motika, Steven Smyth
SYNCHRON '15
SLIC Order

Extended SCCharts

SyncCharts
- Pre
- Signal
- Suspend
- Count Delay

Statecharts
- During Action
  - Complex Final State
- Initialization
- Entry Action
- Connector

SCADE / QUARTZ / Estelle v7
SCCharts Compilation

for ALDO

1: Expand
2: Normalize
(3) Map
SC Graph (SCG)

Circuit-Based Low-Level Synthesis
1: Serialize
2: Split Wires/SSA
Sequentialized SCG
Add Guards

Priority-Based Low-Level Synthesis
Compute Prioritization

Christian Motika, Steven Smyth
SYNCHRON '15
Compiling ALDO (1)

Christian Motika, Steven Smyth

SYNCHRON '15
Compiling ALDO (2)

ALDO
input bool A
bool L = false
output signal D
output bool O = false

[-] Thread1
WaitA
A / L = true
DoneA

[-] Thread2
WaitL
during / D
L / O = true
DoneL

ALDO
input bool A
bool L = false
output bool D
output bool O = false
immediate during / D = false

[-] Thread1
WaitA
A / L = true
DoneA

[-] Thread2
WaitL
during / D = D || true
L / O = true
DoneL

expand signal
Compiling ALDO (3)

ALDO
input bool A
bool L = false
output bool D
output bool O = false
immediate during / D = false

[-] Thread1
WaitA
A / L = true
DoneA

[-] Thread2
WaitL
during / D = D || true
L / O = true
DoneL

expand during action

ALDO
input bool A
bool L = false
output bool D
output bool O = false

[-] _During
I
/ D = false
-S
WaitA
A / L = true
DoneA

[-] Thread2
WaitL
/ D = D || true
L / O = true
DoneL
Compiling ALDO (4)

SCCharts
KIELER Compiler
Usage & Tests
Applications

Christian Motika, Steven Smyth
SYNCHRON '15
Compiling ALDO (5)

Christian Motika, Steven Smyth

SYNCHRON '15
Compiling ALDO (6)

Christian Motika, Steven Smyth

SYNCHRON '15
Compiling ALDO (7)

Christian Motika, Steven Smyth
SYNCHRON '15
Compiling ALDO (8)

Normalization
Compiling ALDO (9)

SCG transformation
Compiling ALDO Demo
Usage

KiCo Compiler (KiCo) 2.0

EOObject

KiCo.UI

Auto Selection
Visual Feedback

Feature Selection

Select All / Expand All

Auto Selection
On/Off

Different Views

Compile
Chains

Christian Motika, Steven Smyth
SYNCHRON '15
**Usage (2)**

### Feature
- Kieler Compiler (KiCo) 2.0
- EObject
- EObject
- EObject

### Transformation
- EObject
- EObject
- EObject

### Processor
- EObject

### KiCo.Server
- HTTP Protocol
- EObject KiCo.compile (EObject, ID1, ID2)

### External Application
- Web
- HTTP request / respond

### Web
- HTTP Protocol

**Steps:**
1. Read request
2. Parse Model
3. Call KiCo.compile()
4. Serialize result
5. Send back result

---

Christian Motika, Steven Smyth
SYNCHRON '15 22
Usage Demo

Christian Motika, Steven Smyth

SYNCHRON '15
Regression Tests
Applications

ABROINO: ABRO SCChart running on Arduino, Dec 2014
More Applications...

Some more student projects... 😊
Proxy States

Application models often contained proxy states that

- accumulate equations
- and were left immediately

Proxy state within the Railway project

Proxy state within the Mindstorms project
Proxy States

Application models often contained proxy states that
• accumulate equations
• and were left immediately

Would be really cool if we could express this in a dataflow way.

However, we don’t want to change the semantics of Core SCCharts!

→ Add a new Extended SCCharts feature!

Proxy state within the Mindstorms project

```plaintext
calculateNewThickness
entry / barThickness = barTickCounter
entry / barTolerance = barThickness * TOLERANCE_BARS_PERCENTAGE / 100
entry / barThicknessMax = barThickness + barTolerance
entry / reader.out(THICKNESS, barThickness)
entry / reader.out(THICKNESS, barThicknessMax)
entry / reader.beep()
```
Recall SCCharts

SCCharts = SyncCharts Syntax + Sequential Constructive Semantics

- Interface declaration
- Region ID
- Transition trigger/effect
- Initial state
- Immediate transition
- Transition priority
- Suspension
- Connector
- Count delay
- History transition
- Conditional termination
- Complex final state

Core-SCCharts
Small set of simple features ease down stream compilation

Extended-SCCharts
Rich set of advanced features ease modeling

Christian Motika, Steven Smyth
SYNCHRON '15
Dataflow Regions

Add dataflow regions as Extended SCCharts feature

**Simple input/output example**

```plaintext
dataflow:
  input bool in;
  output bool out;

out = in;
```

**Simple equation example**

```plaintext
dataflow:
  input int in, in2;
  output int out;

out = (in + in2) * in2;
```

**Node example**

```plaintext
dataflow:
  input int in1, in2;
  output int out;

node add(int a, b)
  returns (int sum) {
    sum = a + b;
  }

call = add(in1, in2);
out = call.sum;
```
Dataflow Regions
Dataflow regions and control-flow regions co-exist

```
dataflow:
    input bool in;
    output bool out;

    node control(bool a) returns (bool o) {
        initial state s1
        -- s2 with a / o = true;
        final state s2;
    }
    call = control(in);
    out = call.o;
```

Defining new node
Containing an control-flow region

Expanded control shows embedded sub-chart

Or simply reference another SCChart

Christian Motika, Steven Smyth
SYNCHRON '15
Dataflow Regions

Dataflow regions and control-flow regions co-exist

Hybrid SCCharts example
Dataflow Transformation

Several approaches are possible to transform the extended feature.

Transform single pass evaluation with initial and final state.

Transform frequent evaluation.

Minimal example:

```plaintext
scchart Minimal {
  input int a, b;
  output int x, y;
  dataflow test:
  x = a + b;
  y = a * b;
}
```

Initialize-Update-Read protocol.
Applications

SCCharts Model Railway Controller Project 2014

STATES:
- 1,628 modeled
- 135,000 expanded
Applications (3)

SCCharts Quadcopter Project 2015

(before the crash)
Applications (4)

SCCharts Lego Mindstorms® Project 2014
To Go Further

CHARLES ANDRÉ.  

GÉRARD BERRY.  
The Esterel v5 Language Primer, 2000.

MOTIKA, C., SMYTH, S., AND VON HANXLEDEN, R.  
Compiling SCCharts – A Case-Study on Interactive Model-Based Compilation.  

SCHNEIDER, C., SPÖNEMANN, M., AND VON HANXLEDEN, R.  
Just model! – Putting automatic synthesis of node-link-diagrams into practice.  
In Proceedings of the IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC’13) (San Jose, CA, USA, 15–19 Sept. 2013).

UNI KIEL, REAL-TIME AND EMBEDDED SYSTEMS GROUP.  
KIELER & SCCharts webpage.  

VON HANXLEDEN, R., LEE, E. A., MOTIKA, C., AND FUHRMANN, H.  
Multi-view modeling and pragmatics in 2020 — position paper on designing complex cyber-physical systems.  

VON HANXLEDEN, R., DUDERSTADT, B., MOTIKA, C., SMYTH, S., MENDLER, M., AGUADO, J., MERCER, S., AND O’BRIEN, O.  
Sequentially Constructive Concurrency—A conservative extension of the synchronous model of computation.  
That’s all Folks - Thank You!
Traditional vs. Interactive SLIC

Textual or Graphical Modeling

Compiler

Code

Textual Modeling

Select Transformation

Compiler

Modeled Diagram

(Intermediate) Transformed Diagram / Code

Christian Motika, Steven Smyth

SYNCHRON '15
# Traditional vs. Interactive SLIC

<table>
<thead>
<tr>
<th>Feature</th>
<th>Traditional</th>
<th>Interactive SLIC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Understand language feature</td>
<td>-</td>
<td>+</td>
</tr>
<tr>
<td>Understand language</td>
<td>-</td>
<td>+</td>
</tr>
<tr>
<td>Compare language features</td>
<td>-</td>
<td>+</td>
</tr>
<tr>
<td>Compare compilation options</td>
<td>-</td>
<td>+</td>
</tr>
<tr>
<td>Fine tuning</td>
<td>-</td>
<td>+</td>
</tr>
<tr>
<td>Choose best suited features</td>
<td>-</td>
<td>+</td>
</tr>
<tr>
<td>Choose best suited transformations</td>
<td>-</td>
<td>+</td>
</tr>
<tr>
<td>Study static feature semantics</td>
<td>-</td>
<td>+</td>
</tr>
<tr>
<td>Study dynamic feature semantics</td>
<td>-</td>
<td>+</td>
</tr>
<tr>
<td>Understanding the models</td>
<td>-</td>
<td>+</td>
</tr>
<tr>
<td>Maintainability</td>
<td>-</td>
<td>+</td>
</tr>
<tr>
<td>Selective validation</td>
<td>-</td>
<td>+</td>
</tr>
<tr>
<td>Isolated error fixes</td>
<td>-</td>
<td>+</td>
</tr>
<tr>
<td>Extending the language/compiler</td>
<td>-</td>
<td>+</td>
</tr>
<tr>
<td>Performance</td>
<td>+</td>
<td>+/-</td>
</tr>
</tbody>
</table>
SCCharts Modeling User Story

1. Edit SCT code
2. Select transformations
3. Inspect original + transformed SCChart
4. Adjust layout
# Mapping SCCharts to SCG

<table>
<thead>
<tr>
<th></th>
<th>Region (Thread)</th>
<th>Superstate (Parallel)</th>
<th>Trigger (Conditional)</th>
<th>Action (Assignment)</th>
<th>State (Delay)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Normalized SCCharts</strong></td>
<td><img src="image1" alt="Diagram" /></td>
<td><img src="image2" alt="Diagram" /></td>
<td><img src="image3" alt="Diagram" /></td>
<td><img src="image4" alt="Diagram" /></td>
<td><img src="image5" alt="Diagram" /></td>
</tr>
<tr>
<td></td>
<td><img src="image6" alt="Diagram" /></td>
<td><img src="image7" alt="Diagram" /></td>
<td><img src="image8" alt="Diagram" /></td>
<td><img src="image9" alt="Diagram" /></td>
<td><img src="image10" alt="Diagram" /></td>
</tr>
<tr>
<td><strong>SCG</strong></td>
<td><img src="image11" alt="Diagram" /></td>
<td><img src="image12" alt="Diagram" /></td>
<td><img src="image13" alt="Diagram" /></td>
<td><img src="image14" alt="Diagram" /></td>
<td><img src="image15" alt="Diagram" /></td>
</tr>
</tbody>
</table>
KiCo Selection Algorithm

Kieler Compiler (KiCo) 2.0 – Advanced Transformation Selection

- User selected
- Auto selected
- Auto selected but not processed
- Expanded by
- Produces
- Not handled by
- Feature contained in model
- Feature not contained in model

Christian Motika, Steven Smyth
SYNCHRON '15
BAK 5
Test File Creation (ESO)

1: Inspect model
2: Load model

Model

ESO Generator

Reference Compiler

Manually write reasonable ESO file traces

Update ESO with expected outputs

Stimulate running model with generated random inputs

Generate random inputs

Automatically synthesized ESO file

Manually created ESO file

Christian Motika, Steven Smyth
SYNCHRON '15
Model Railway Track Scheme

Track segments & Directions:
- Inner Circle
- Outer Circle
- Kicking Horse Pass
- Interconnections
  - Unidirectional block
  - Bidirectional block with forward direction
  - Preferred direction

Electronics:
- Block isolation
- Reed contact
- Block signal
- Lighting
- Point operating stall

Track specialties:
- Bridge
- Point or crossing
- Railroad crossing
SCCharts Meta Model

- Emission
- ValuedObject
- Assignment
- TextEffect
- Scope
- Substitution

StateType:
- NORMAL
- REFERENCE
- TEXTUAL

TransitionType:
- WEAKABORT
- STRONGABORT
- TERMINATION

Expression:
- from expressions

Emission:
- 0..1 newValue

ValuedObject:
- from expressions

Assignment:
- 1 expression

TextEffect:
- 1

Scope:
- id: EString
- label: EString

Substitution:
- formal: EString
- actual: EString

State:
- type: StateType

Region:
- parentRegion: 0..1

Transition:
- priority: EInt
- type: TransitionType

IncomingTransitions:
- targetState: 0..*

OutgoingTransitions:
- sourceState: 0..*

parentState:
- 0..1

renamings:
- 0..*