A Model-driven development and verification framework for embedded software (3)

Yunja Choi

Software Safety Engineering Laboratory
School of Electrical Engineering and Computer Science
Kyungpook National University
Work in progress

- Model-driven and component-based design verification
  - Supporting tool for design verification
  - Performance improvement using component control model

- Safety analysis of OSEK/VDX for automobile software
Model-driven and component-based design verification
Model-driven and component-based design verification

Specification
- Statecharts
- Class Diagram
- Sequence Diagram
- Operation Schemata
- Object Diagram (Architecture)

Realization
- Kpt A
- Kpt B
- Kpt C
- Kpt D

Architectural Realization: Kpt A → Kpt B → Kpt C → Kpt D
Example

1st level abstract component

2nd level abstract component

3rd level abstract component
Supporting tools
Construction of abstract behavior (1)

(a) two dependent abstract components

(b) free composition

(c) synchronized reduction

(d) abstraction
Construction of abstract behavior (2)
Checking communication consistency
## Performance

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Depth</th>
<th>States</th>
<th>Transitions</th>
<th>Memory (M bytes)</th>
<th>Time (Seconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Msp430Timer32khzC</td>
<td>realization</td>
<td>1,494,716</td>
<td>2.4e+07</td>
<td>1.22e+08</td>
<td>5,034.0</td>
<td>1.69e+03</td>
</tr>
<tr>
<td></td>
<td>specification</td>
<td>392,826</td>
<td>8e+06</td>
<td>2.92e+07</td>
<td>843.5</td>
<td>224</td>
</tr>
<tr>
<td>Alarm32khz16C</td>
<td>realization</td>
<td>9,047</td>
<td>4.3e+07</td>
<td>2.02e+08</td>
<td>14,492.5</td>
<td>3.97e+03</td>
</tr>
<tr>
<td></td>
<td>specification</td>
<td>761</td>
<td>283,461</td>
<td>438,704</td>
<td>15.5</td>
<td>2.44</td>
</tr>
</tbody>
</table>
Issues

- Abstraction is abstraction
  - Timing and communication-related issues are ignored
- Still need compositions of components in the same abstraction level
  - A systematic method for the reuse of components is desired
Explicit modeling for synchronous calls can reduce verification complexity while maintaining the behavior of individual components.
Control models for synchronous call

- Synchronous call to external services activates external components and deactivates itself until its return message arrives
- Assumption: one processor model
Control models for synchronous call
Control models for synchronous call

stub1
stub2
stub3

environment

Synchronous event/consume & return
Wait for events
asynchronous event/consume

external event(activator.pid)
processed event

Done/activate activator
Wait for return

Return control
Wait for control
/event generation; activate target component
active

external service;
/CALL external service;
Wait for return

inactive
done/activate activator
Control models for synchronous call

alarm?start, dt/alarm!RV, current_time;
interruptPending=FALSE;
events_enabled=TRUE;

alarm?isRunning/alarm!RV, events_enabled;
alarm?stop/events_enabled=FALSE;

inactive

activate

Wait for event

Event processing 2

/alarm!RV, events_enabled;
Activate environment;

alarm?isRunning

Event processing 1

/alarm!RV, current_time;
interruptPending=FALSE;
events_enabled=TRUE;
activate environment

activated

No event/activate environment

wait for event

/
Verification framework
## Experiments

<table>
<thead>
<tr>
<th>name</th>
<th>type</th>
<th>depth</th>
<th>states</th>
<th>transitions</th>
<th>Memory</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>E</td>
<td>unit</td>
<td>1,302</td>
<td>1,891,489</td>
<td>2,366,210</td>
<td>112</td>
<td>11</td>
</tr>
<tr>
<td>D</td>
<td>unit</td>
<td>8,361</td>
<td>162,844</td>
<td>2.39E+05</td>
<td>8.85</td>
<td>0.17</td>
</tr>
<tr>
<td>B</td>
<td>composition</td>
<td>214,157</td>
<td>8,605,951</td>
<td>1.33E+07</td>
<td>819.5</td>
<td>16.3</td>
</tr>
<tr>
<td>A</td>
<td>composition</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
</tr>
</tbody>
</table>

<models without consideration of synchronous calls >

<table>
<thead>
<tr>
<th>name</th>
<th>type</th>
<th>Depth</th>
<th>states</th>
<th>transitions</th>
<th>Memory</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>E</td>
<td>Unit</td>
<td>2,715</td>
<td>101,694</td>
<td>135,115</td>
<td>95.7</td>
<td>0.35</td>
</tr>
<tr>
<td>D</td>
<td>Unit</td>
<td>91</td>
<td>758</td>
<td>1.14E+03</td>
<td>2.5</td>
<td>0.001</td>
</tr>
<tr>
<td>B</td>
<td>Composition</td>
<td>763</td>
<td>12,227</td>
<td>1.94E+04</td>
<td>3.6</td>
<td>0.018</td>
</tr>
<tr>
<td>A</td>
<td>Composition</td>
<td>43,436</td>
<td>268,819</td>
<td>388.931</td>
<td>83.0</td>
<td>0.43</td>
</tr>
</tbody>
</table>

< models composed with the control models for synchronous calls >
Next step

- Need to update the translator for full automation
- more extensive experiments with TinyOS
- Application to OSEK/VDX
  - Model-based safety analysis using model checking