Esterel Studio

Update

Kim Sunesen
Esterel EDA Technologies

www.estrel-eda.com

Synchron, November 2007, Bamberg Germany
Agenda

- Update on Esterel Studio
- Architecture Diagrams
- Formal Verification
- IEEE standardization
- Early Performance Estimation
Industries Served

Esterel Studio™
Front-end design and verification suite for control-intensive hardware
- Rigorous & unambiguous executable specifications
- ESL synthesis path to efficient RTL / SystemC code
- Formal Verification

SCADE Suite™
De-facto Standard for Safety-critical avionics embedded software
- DO-178B Level A certified systems
- Automatically-generated C code

SCADE Drive™
Safety-critical automotive embedded software
- Code generator certified by TUV - IEC 61508 standard
Esterel Customer Consortium

- Best practice sharing
- Design flow integration
- Collaborative roadmap
- Implemented on projects
- **First Silicon 2005**
Esterel Consortium Industrial Designs

- 50+ industrial designs, some with ECOs
- Interoperability with 16 EDA tools from ESL to DFT

- AHB, APB, OPB, OCP, AXI bus bridges, interfaces, converter, transactors
- Cache coherence implementation and verification
- Processor architecture validation
- Processor & co-processor
- Traffic controller, SD/MMC Card controller
- Memory architecture
- External Memory controller
- Serial ATA
- DMAs
- Power management
- Video streaming
- Battery controller
- UMTS/GPRS protocol specification and verification
- Smart Cards security formal verification
- Video controller
Esterel Studio: the Front-end Design and Verification Suite

**Design Specification Capture**

- **Architecture**
- **Project Structure**
- **Automatic Documentation**
- **Executable Specification Exporter**
- **Functional Design Spec**
- **Verification Requirements**
- **Debugging & Simulation**
- **Formal Verification**
- **Design Verification**
  - Sequential Equivalence check

**Integrated Development Environment**

- **Project Management**
  - Block editor
  - redesigned

- **Code Generation**
  - early perf estimation
  - modular multi-clock

**SystemC**

- Optimized for synthesis
- DFT-ready
- SystemC & RTL flow integrated

**RTL Synthesis**

- new in 6.0
- new strategies
- Functional Design Spec
- Verification Requirements
- Debugging & Simulation
- Formal Verification
- Sequential Equivalence check

**New Technologies**

- Functional Design Specification
- Code Coverage
- Early Performance Estimation
- Block Editor

**Integration Points**

- SystemC
- .sc
- IP-XACT
- .vhd
- RTL Synthesis
Architecture Diagrams
Editor

- Fully integrated in the workspace (navigation, tree, menu)
- Dedicated view
- Architecture diagram modules are hierarchical like SSM
Architecture Diagram Components

Connection Bar
Textual Instance
SSM Instance
Connection point
Output Ports
Hierarchical Architecture Diagram

Local Port
Hierarchical Architecture Diagram
Auto-asserts are automatically extracted from the design

Auto-asserts prevent common mistakes
- Overflow errors, Array index errors, Division by zero
- Reading uninitialized signals before writing them
- Multiple emissions (write / write races)
Comparing execution time for Standard and Interpolation strategies

- Standard strategy verifies very quickly "simple" properties
- Interpolation more effective on "complex" properties
Esterel language IEEE standardization

- DASC (Digital Automation Standards Committee) and IEEE group started standardizing the Esterel language
  - Language is public and Language Reference Manual will become property of IEEE
  - started March 2007

- Industry participants
  - General Motors, IBM, Intel, Microsoft, NEC, NXP, Orange, STMicroelectronics, Synopsys, Texas Instruments

- Academic participants
  - Columbia Univ., E.I. Geneva, INRIA, Kaiserlautern Univ., Technion Israel, Tübingen Univ, etc.
Current Hot Subjects

- Data definition and data expressions
  - `struct`, tuples
  - simplified u2bin / bin2u conversions
  - functional module calls

- Signals, Interfaces, and modules
  - declaration simplification (`temp value` the default)
  - `struct` = values of ports
  - port emission by struct assignment
  - module behavior extension (= code in interfaces)

- Genericity
  - Improved and generalized mechanism
Early Performance Estimation

The Problem
- From a high-level Esterel model, designers wish to
  - Identify created operators
  - Evaluate the size and speed of the resulting circuit,
- But running synthesis is too costly
  - Slows down the optimization loop
- Quick approximate estimation is needed!

The Solution
- Same type of information as synthesis:
  - actually allocated registers
  - estimated area consumption
  - max estimated delay path
  - operator count
- Estimation is quicker than synthesis (> 6x)
- Tracing back costly constructs is made easier
**HTML Screenshots (1/3)**

### Estimation Summary

<table>
<thead>
<tr>
<th>Combinational Area</th>
<th>Max Delay</th>
<th>Register</th>
</tr>
</thead>
<tbody>
<tr>
<td>3112.14</td>
<td>38.04</td>
<td>79</td>
</tr>
</tbody>
</table>

- General summary for the entire design

### Max Delay Path

**Startpoint:** register PauseReg_WaitForWord_38_1_L1D1  
**Endpoint:** output OutPixel_data

<table>
<thead>
<tr>
<th>Point</th>
<th>Module</th>
<th>Incr</th>
<th>Path</th>
</tr>
</thead>
<tbody>
<tr>
<td>PauseReg_WaitForWord_38_1_L1D1</td>
<td>Feeder</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>Then1_7_1_L1D1_and_x_506_507</td>
<td>Feeder</td>
<td>1.08</td>
<td>1.08</td>
</tr>
<tr>
<td>Status_InPixel_S8_0_or_x_602_603</td>
<td>Filter</td>
<td>2.16</td>
<td>3.24</td>
</tr>
<tr>
<td>Then2_13_2_and_604_605</td>
<td>PixelFilter</td>
<td>0.95</td>
<td>4.19</td>
</tr>
<tr>
<td>sv_DelayLine_V11[0]_mux_607</td>
<td>PixelFilter</td>
<td>1.00</td>
<td>5.19</td>
</tr>
<tr>
<td>factor_685</td>
<td>PixelFilter</td>
<td>11.00</td>
<td>16.19</td>
</tr>
<tr>
<td>sv_Product_V9[0]_mux_686</td>
<td>PixelFilter</td>
<td>1.00</td>
<td>17.19</td>
</tr>
</tbody>
</table>

- Max estimated delay path
- With source module info
Input - Output Combinational Paths

<table>
<thead>
<tr>
<th>Output</th>
<th>Inputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>OutPixel_data</td>
<td>(InWord, InWord_data[31:0])</td>
</tr>
<tr>
<td>Ready</td>
<td>(InWord)</td>
</tr>
<tr>
<td>OutPixel</td>
<td>(InWord)</td>
</tr>
<tr>
<td>OutEndOfLine</td>
<td>(InWord)</td>
</tr>
</tbody>
</table>

Tells whether design is Mealy vs. Moore

Module List

<table>
<thead>
<tr>
<th>Module</th>
<th>Instanciated</th>
<th>Area</th>
</tr>
</thead>
<tbody>
<tr>
<td>Filter</td>
<td>1</td>
<td>45</td>
</tr>
<tr>
<td>Feeder</td>
<td>1</td>
<td>99</td>
</tr>
<tr>
<td>PixelFilter</td>
<td>1</td>
<td>2968.14</td>
</tr>
</tbody>
</table>

Module instance sizes
### Operator Count

#### Bool operators

<table>
<thead>
<tr>
<th>Operator</th>
<th>BitWidth1</th>
<th>BitWidth2</th>
<th>Area</th>
<th>Number</th>
<th>Total Area</th>
</tr>
</thead>
<tbody>
<tr>
<td>or</td>
<td>5</td>
<td></td>
<td>5.00</td>
<td>7</td>
<td>35.00</td>
</tr>
<tr>
<td>and</td>
<td>3</td>
<td></td>
<td>2.00</td>
<td>44</td>
<td>88.00</td>
</tr>
<tr>
<td>dszdyni_1</td>
<td>2</td>
<td></td>
<td>1.00</td>
<td>38</td>
<td>38.00</td>
</tr>
</tbody>
</table>

#### Unsigned operators

<table>
<thead>
<tr>
<th>Operator</th>
<th>BitWidth1</th>
<th>BitWidth2</th>
<th>Area</th>
<th>Number</th>
<th>Total Area</th>
</tr>
</thead>
<tbody>
<tr>
<td>factor</td>
<td>8</td>
<td>3</td>
<td>406.45</td>
<td>5</td>
<td>2032.25</td>
</tr>
<tr>
<td>add</td>
<td>12</td>
<td>11</td>
<td>359.95</td>
<td>1</td>
<td>359.95</td>
</tr>
<tr>
<td>add</td>
<td>11</td>
<td>11</td>
<td>342.47</td>
<td>1</td>
<td>342.47</td>
</tr>
</tbody>
</table>

Operators count sorted by signal source types
How Early Perf is calibrated

1. **Measures:** every operator was synthesized with a variable bit-size and fanin

2. **Calibration:** using those previous synthesis result, one determines an operator sizing formula or table

<table>
<thead>
<tr>
<th>Fanin</th>
<th>Area</th>
<th>Perf Area</th>
<th>Av_Error%</th>
<th>Delay</th>
<th>Perf Delay</th>
<th>Av_Error%</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>2</td>
<td>2</td>
<td>8.9</td>
<td>0.87</td>
<td>0.95</td>
<td>8.77</td>
</tr>
<tr>
<td>8</td>
<td>5</td>
<td>5</td>
<td>1.4</td>
<td>1.75</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>8</td>
<td>7.47</td>
<td>2.07</td>
<td>2.02</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>9</td>
<td>8.81</td>
<td>2.24</td>
<td>2.29</td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>10</td>
<td>11.51</td>
<td>2.18</td>
<td>2.53</td>
<td></td>
<td></td>
</tr>
<tr>
<td>24</td>
<td>18</td>
<td>16.9</td>
<td>3.01</td>
<td>2.66</td>
<td></td>
<td></td>
</tr>
<tr>
<td>32</td>
<td>38</td>
<td>22.3</td>
<td>2.53</td>
<td>2.79</td>
<td></td>
<td></td>
</tr>
<tr>
<td>40</td>
<td>35</td>
<td>38.58</td>
<td>3.01</td>
<td>2.92</td>
<td></td>
<td></td>
</tr>
<tr>
<td>48</td>
<td>46</td>
<td>40.68</td>
<td>3.33</td>
<td>3.06</td>
<td></td>
<td></td>
</tr>
<tr>
<td>56</td>
<td>43</td>
<td>42.77</td>
<td>3.12</td>
<td>3.19</td>
<td></td>
<td></td>
</tr>
<tr>
<td>64</td>
<td>43</td>
<td>44.87</td>
<td>3.12</td>
<td>3.32</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Results on Industrial designs

<table>
<thead>
<tr>
<th>Design type</th>
<th>Area (syn)</th>
<th>Area (perf)</th>
<th>Area error rate</th>
<th>Speed (syn)</th>
<th>Speed (perf)</th>
<th>Speed error rate</th>
<th>time gain</th>
</tr>
</thead>
<tbody>
<tr>
<td>Write posting bus module</td>
<td>9394</td>
<td>11744</td>
<td>25%</td>
<td>18.42</td>
<td>20.6</td>
<td>12%</td>
<td>7x faster</td>
</tr>
<tr>
<td>Line filter</td>
<td>9235</td>
<td>10694</td>
<td>16%</td>
<td>47.2</td>
<td>51.77</td>
<td>10%</td>
<td>57x faster</td>
</tr>
<tr>
<td>DMA 1</td>
<td>8515</td>
<td>14387</td>
<td>69%</td>
<td>22.87</td>
<td>23.06</td>
<td>1%</td>
<td>12x faster</td>
</tr>
<tr>
<td>Transactor AXI</td>
<td>34757</td>
<td>69810</td>
<td>101%</td>
<td>44.22</td>
<td>72.83</td>
<td>65%</td>
<td>6x faster</td>
</tr>
</tbody>
</table>
Technology & Timing dependency

- Current areas / frequencies are based on the lsi_10k library
- They can be adapted to other libraries by re-calibration

- Remark: early performance estimation does not consider timing constraints
Thank you!