High-Level Development Tools

Data Flow  C Code  Textual Math  Modeling  Statechart

EE249 Lab
September 30th, 2008
Hugo A. Andrade

Graphical System Design Platform

LabVIEW™

Desktop Platform

Linux®  Macintosh  Windows

Embedded Platform

Real-Time  FPGA  Micro

ni.com/statechart
Virtual Instrumentation

• Hardware and Driver Software

• Application Software

• Network
Virtual Instrumentation
Graphical System Design Platform
Graphical System Design Platform
The G (LabVIEW) Language Model

• Homogenous dataflow language
  ▪ Structured case (switch, select) and loops
    • “Structured dataflow”
• Run-time scheduling
  ▪ Explicit task level parallelism
  ▪ Implicit parallelism heuristically identified
• Synthesizable language
  ▪ To machine code on x86 and PPC processors
  ▪ To VHDL for FPGAs
  ▪ To C for embedded processors
• Turing complete
Evolution of LabVIEW Backend Technologies

- Intermediate Code: None or (Machine Code)
- Compiler: LabVIEW, OEM Synthesis (PAR)
- Hardware Target: Wintel, PowerPC, FPGA, DSP, Any 32-bit, MPU

Diagram:
- VHDL
- C
- None (Object Library)
- LabVIEW
- Any
What is LabVIEW FPGA

- LabVIEW FPGA Module
- Reconfigurable I/O (RIO) Hardware
Enforcing Dataflow in FPGA
Developing Applications with the NI LabVIEW Statechart Module
High-Level Development Tools

Data Flow

C Code

Textual Math

Modeling

Statechart

Graphical System Design Platform

LabVIEW™

Desktop Platform

Linux®, Macintosh, Windows

Embedded Platform

Real-Time, FPGA, Micro

ni.com/statechart
Statecharts are visual representations of reactive (event-based) systems.
Differences between Statecharts and FSMs

Both contain the same basic concepts:
- States
- Transitions

Statechart adds additional concepts:
- Hierarchy
- Concurrency
- Event-based paradigm
- Pseudostates & Connectors

Based on the UML statechart diagram specification
Reactive Systems

• Communication systems
• Digital protocols
• Control applications
  ▪ Sequential logic
  ▪ Batch processing
  ▪ Event response
  ▪ Non-linear control
• User-interface implementation
• System modeling for virtual prototyping (simulation)
Statechart Benefits

• Abstraction
  ▪ Simple semantics to represent complex systems
  ▪ System-level view
  ▪ Self-documenting
Machine & Process Control

hierarchy

concurrency
FPGA Logic

Send Bits

Idle

Set CS

Set Clk

Reset Clk & Output

Reset CS

hierarchy

ni.com/statechart

NATIONAL INSTRUMENTS
User Interfaces

LabVIEW Statechart Module

Color Cycle State

- Red State
- Blue State
- Green State

White State

history

ni.com/statechart

NATIONAL INSTRUMENTS
Statechart Benefits

• Abstraction
  ▪ Simple semantics to represent complex systems
  ▪ System-level view
  ▪ Self-documenting

• Scalability
  ▪ Easily extend applications
  ▪ Open software platform

• Automatic Code Generation
  ▪ LabVIEW Embedded Technology
LabVIEW Statechart Development

1. Build statechart
2. Define transitions and states
3. Generate statechart subVI
4. Place in LabVIEW block diagram
Example – Ceiling Fan

- **Triggers**
  - Power switch
  - Fan toggle
  - Light toggle

- **Outputs**
  - Light
  - Fan speed

<table>
<thead>
<tr>
<th>Power</th>
<th>No Power</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fan</td>
<td>Light</td>
</tr>
<tr>
<td>High</td>
<td>on</td>
</tr>
<tr>
<td>medium</td>
<td>off</td>
</tr>
<tr>
<td>low</td>
<td>off</td>
</tr>
<tr>
<td>off</td>
<td></td>
</tr>
</tbody>
</table>
Example – Ceiling Fan

- **Triggers**
  - Power switch
  - Fan toggle
  - Light toggle

- **Outputs**
  - Light
  - Fan speed

- **Internal Data**
  - Fan Speed

---

<table>
<thead>
<tr>
<th>Power</th>
<th>No Power</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>Fan on</td>
<td>Light on</td>
</tr>
<tr>
<td>Fan off</td>
<td>Light off</td>
</tr>
</tbody>
</table>

Source: ni.com/statechart
1. Build Statechart
1. Build Statechart
1. Build Statechart
2. Define Transitions and States

• Each Transition contains three components
  - **Trigger** – events that cause a transition
  - **Guard** – logic that can prevent a transition
  - **Action** – what happens when you transition

- **Curr State** – DOOR CLOSED
  - **Trigger** – doorbell ring
  - **Guard** – adult home?
  - **Action** – open door
  - **New State** – DOOR OPEN

*If the doorbell rings and an adult is home, answer the door.*
2. Define Transitions and States

- Each Transition contains three components
  - **Trigger** – events that cause a transition
  - **Guard** – logic that can prevent a transition
  - **Action** – what happens when you transition

- Each state contains three types of actions
  - **Entry** – what happens when you get there
  - **Exit** – what happens when you leave
  - **Static** – what happens while you are there
2. Define **Transitions and States**

**Trigger-Guard-Action**

**Triggers**
2. Define Transitions and States

Static Reaction

Trigger-Guard-Action

Inputs

State Data

Outputs
3. Build Statechart SubVI
4. Place in LabVIEW Block Diagram

Asynchronous Usage
- User interface
- Interruption handling
- Modeling event driven systems
4. Place in LabVIEW Block Diagram

Synchronous Usage
- Embedded applications
- Communication protocols
- Control implementations
Statechart Execution

• Evaluate the trigger/guard logic for the transitions leaving the current state(s)

• On first valid transition:
  ▪ Execute the exit action(s) for the current state(s)
  ▪ Execute the transition action
  ▪ Execute the entry action(s) for all state(s) being transitioned to

• If no transitions are valid:
  ▪ Evaluate the trigger/guard logic for all static reactions configured for the current state
  ▪ Execute the action code for all valid reactions
Statechart Execution

A. Check transitions.
   - Are there unprocessed transitions?
     - Yes → Does this transition have a trigger?
       - Yes → Does the incoming trigger match the one you specified for this transition?
         - Yes → Execute transition action.
         - No → Move to the next state.
       - No → Execute entry action.
     - No → Transition is invalid.
   - No → There are no valid transitions.

B. Handle actions.
   - Is there a state exit action?
     - Yes → Execute state exit action.
     - No → Is there a transition action?
       - Yes → Execute transition action.
       - No → Does this state have an entry action?
         - Yes → Execute entry action.
         - No → Check internal queue.

C. Handle static reactions.
   - Are there unprocessed reactions?
     - Yes → Does the next reaction have a trigger?
       - Yes → Does this incoming trigger match the one you specified for this reaction?
         - Yes → Does this reaction have a guard?
           - Yes → Does the guard return TRUE or FALSE?
             - TRUE → Check an action.
             - FALSE → Internal queue.
           - No → Execute action.
         - No → Internal queue.
     - No → Internal queue.
DEMO
What to do next?

• Visit ni.com/statechart
  ▪ Demo videos
  ▪ Statecharts 101 whitepaper
  ▪ Statecharts with LabVIEW FPGA whitepaper
  ▪ Try the LabVIEW Statechart Module online

• Demonstration from local Field Engineer