Stateflow for Signal Processing and Communications Applications with Code Generation Capabilities

MathWorks Korea

이웅재 부장
Senior Application Engineer
Agenda

- State Machines in Signal Processing & Communication Systems
- Overview of State Machine and Flow Graph
- Comparison between Different Approaches
- Designing State Machines using Model-Based Design
- Automatic Code Generation from State Machines
  - Generating C code
  - Generating HDL code
- Additional Resources
Example: WWV Digital Time-Code Receiver
Agenda

State Machines in Signal Processing & Communication Systems

- Overview of State Machine and Flow Graph
- Comparison between Different Approaches
- Designing State Machines using Model-Based Design
- Automatic Code Generation from State Machines
  - Generating C code
  - Generating HDL code
- Additional Resources
State Machine in Signal Processing & Communication Applications

Defense Industry
- ECM, ELINT, Digital Receiver

Communication Network System
- Router, Ethernet

Semiconductor
- Microprocessor, SoC, Memories

Consumer Electronics
- Rice Cooker, Washing Machine

Industry Machineries
- SMT Machine, PCB Tester, Component Tester
Algorithms and Protocols: Signal Processing and Communications

- Mode/Supervisory Logic
- Dynamic Data/Signal Path Control
- System Controller
- Signal Acquisition and Tracking
- Call Processing
- Protocols Control
- Acknowledgement Schemes (Automatic Repeat Request)
- Event Driven or Reactive Algorithms
- Traffic Modeling

For these algorithms, a finite state machine paradigm is more appropriate than a signal flow paradigm
Example Application: Video Processing

- Supervisory control logic
- Increased processing efficiency
- Advanced visualization
Example Application: Bluetooth Transmitter

- Bluetooth 2.0 with enhanced data rate (EDR)
- Multiple adaptive modes
- Scheduling of different signal processing algorithms
Agenda

- State Machines in Signal Processing & Communication Systems

Overview of State Machine and Flow Graph

- Comparison between Different Approaches
- Designing State Machines using Model-Based Design
- Automatic Code Generation from State Machines
  - Generating C code
  - Generating HDL code
- Additional Resources
What Is a State Machine?

- Represent reactive systems that have states or modes
- States change based on defined conditions and events
- It can only behave in a predefined number of ways (states)

![State Machine Diagram](image-url)
What are Mealy and Moore machines?

### Mealy
- All actions are condition actions
- Output function of state and input

### Moore
- All actions are state actions
- Output function of state only
What is a Flow Graph?

- It represents an algorithm or process like a flow chart.
- It is used primarily for modeling stateless logic, where the flow graph maintains no memory of previous inputs or outputs.
- It can be used to model control flows like “if-else” condition, “for-loop”, and “while-loop”
What is Stateflow®? : SPC Case

- Model and simulate decision logic for reactive systems:
  - supervisory/mode control
  - task scheduling
  - dynamic data path control
  - protocol control
  - reactive/event driven algorithm

- Develop mode-logic using state machines and flow charts

- Provide diagram animation and integrated debugger
How does Stateflow® work with Simulink®?

Simulink is used to respond to **continuous** changes in dynamic systems.

Stateflow is used to respond to **instantaneous** changes in dynamic systems.

Real-world systems have to respond to both continuous and instantaneous changes.

 suspension dynamics
gear changes

 propulsion system
liftoff stages

 robot kinematics
operation modes

*Use both Simulink and Stateflow so that you can use the right tool for the right job.*
Agenda

- State Machines in Signal Processing & Communication Systems
- Overview of State Machine and Flow Graph

Comparison between Different Approaches

- Designing State Machines using Model-Based Design
- Automatic Code Generation from State Machines
  - Generating C code
  - Generating HDL code
- Additional Resources
Different way of designing State Machine

Provide a natural, graphical language for modeling state machines

State chart

Simulink blocks

MATLAB Function

```matlab
function y = fcn(u)
    persistent x;
    if isempty(x)
        x = 1;
    end
    if (x == 1)
        if (u > 5)
            x = 2;
        end
    else
        if (u < -5)
            x = 1;
        end
    end
    if (x == 1)
        y = 0;
    else
        y = 1;
    end
end
```
Other Approaches for Modeling Control Logic

```c
void a_step(void)
{
    if (a_DWork.is_active_cl_a == ON) {
        a_DWork.is_active_cl_a = OFF;
        if (a_DWork.is_active_cl_a == IN_passive) {
            a_DWork.is_cl_a = a_IN_passive;
            a_y.LO_mode = Passive;
        } else {
            switch (a_DWork.is_cl_a) {
            case a_IN_active:
                if (!a_DWork.RO_act && !a_DWork.LI_act) {
                    a_DWork.is_cl_a = a_IN_standby;
                    a_y.LO_mode = Standby;
                }
                break;
            case a_IN_passive:
                if (a_DWork.LI_act == 0.0) {
                    a_DWork.is_cl_a = a_IN_standby;
                    a_y.LO_mode = Standby;
                } else {
                    if (!a_DWork.LI_act || !a_DWork.RO_act) {
                        a_DWork.is_cl_a = a_IN_active;
                        a_y.LO_mode = Active;
                    }
                }
                break;
            case a_IN_standby:
                if (!a_DWork.LI_act || a_DWork.RO_act == 0.0) {
                    a_DWork.is_cl_a = a_IN_active;
                    a_y.LO_mode = Active;
                }
                break;
            default:
                a_DWork.is_cl_a = a_IN_NO_ACTIVE_CHILD;
                break;
            }
        }
    }
}
```

Simulink Diagram
Actuators
en:L_switch(); R_switch();
du:L_switch(); R_switch();

truthtable
L_switch
R_switch

function
y_act = LO_act
y_act = LI_act
y_act = RO_act
y_act = RI_act
Other Approaches for Modeling Control Logic

Stateflow

C-code

>1000 lines
Agenda

- State Machines in Signal Processing & Communication Systems
- Overview of State Machine and Flow Graph
- Comparison between Different Approaches

Designing State Machines using Model-Based Design

- Automatic Code Generation from State Machines
  - Generating C code
  - Generating HDL code
- Additional Resources
Designing State Machine: Demo

- National Institute of Standards and Technology (NIST) Time Decoder

- Radio Station WWV in Boulder, CO
  - Broadcasts frequency reference standards and time code information
  - Referenced to atomic time scales at NIST
Symbol Synchronization

Search

**Goal:** Establish synchronization (stringent)

**Algorithm:** Search for successive edges with ~1 sec separation

Symbols

Leading Edges

Search Window

Reset on false edges
Symbol Synchronization
Concept Review

- **STATES** represent modes of operation
  - Exclusive States (OR)
  - Parallel States (AND)

- **TRANSITIONS** represent paths between states

- **FLOW GRAPHS** represent complex time and condition based logic

- Types of functions
  - Graphical
  - MATLAB
  - Simulink
  - Truth Tables
Agenda

- State Machines in Signal Processing & Communication Systems
- Overview of State Machine and Flow Graph
- Comparison between Different Approaches
- Designing State Machines using Model-Based Design

Automatic Code Generation from State Machines

- Generating C code
- Generating HDL code

- Additional Resources
Code Generation for Stateflow
Demo: Sequence Detector
Stateflow guidelines for HDL code generation

- Beware unsupported software constructs
- Prefer fixed point types to integer types
- Consider Mealy and Moore state machines when generating HDL code
- Enable the chart property Execute (enter) Chart at Initialization
- Parallel states do not imply concurrency
- Specify a fixed-point constant indirectly in action language by using a fixed-point context-sensitive constant
Agenda

- State Machines in Signal Processing & Communication Systems
- Overview of State Machine and Flow Graph
- Comparison between Different Approaches
- Designing State Machines using Model-Based Design
- Automatic Code Generation from State Machines
  - Generating C code
  - Generating HDL code

Additional Resources
Additional Resources

- Product Web pages:
  www.mathworks.com/products/stateflow/
  http://www.mathworks.com/products/embedded-coder/
  http://www.mathworks.com/products/hdl-coder/

- Signal Processing and Communications Web page:
  www.mathworks.com/applications/dsp_comm/

- Webinars
  Recorded: www.mathworks.com/company/events/archived_webinars.html
  Upcoming: www.mathworks.com/company/events/webinars/upcoming.html

- MathWorks Training
  www.mathworks.com/services/training/

- Contact your sales rep!