Lecture 4:
System Specification

Sharif University of Technology
Computer Engineering Dept.

Winter-Spring 2008

Mehdi Modarressi
Topics for Today

- System specification
- Models of computation
- Specification languages
Embedded systems are becoming more and more complex

Increasing the abstraction level
- Reducing the objects designers have to manage
- Shift the complexity to the CAD tools

System-level design
- Hardware-software co-design
System Specification

- Describes the functionality of the system without specifying the implementation

- May be *executable* to allow dynamic verification

- Two fundamental concepts:
  - Models of computation
  - Languages
Model of Computation (MOC)

- Model implies a way of thinking
- Model is a method and rules to:
  - Think the system as some sub-systems or pieces
  - Composing the pieces to create system functionality
- Model should be
  - Formal => avoid ambiguity
  - Complete => can describe entire system
  - Comprehensible and easy-to-modify
Models

- **Model vs. Language**
  - We use a *model* to decompose a system into pieces (Model implies a way of thinking)
  - We generate a specification by describing these pieces in a particular *language* (Language is a tool for description)

- Certain languages better at capturing certain computation models
Models of Computation

- Von-Neumann model
- Discrete-event model
- CFSM model
- Data-flow model
- ....
Von-Neumann MoC

- Von-Neumann MoC is different from Von-Neumann architecture
- Sequential computing
- No notation of time
- No synchronization and communication protocols
- Example language: C, Pascal
Discrete-Event Model

- The MoC of HDLs: Verilog, VHDL
- Based on a queue of scheduled events
  - In a cyclic manner:
    - Read the event at the top of the queue
    - Change the system variables and states based on the event
    - Increase the time
    - Add the generated events to the queue
Finite-State Machine Model (FSM)

- A set of *states* and a set of *transitions* between them
  - The system is composed of some states
  - At each clock tick, system checks the inputs from the environment
  - If the inputs are changed, system processes the input and
    - Generate output
    - Change the state

- Most popular model for control systems
  - reactive
Formal Definition

An FSM is a 6-tuple $<S, I, O, F, H, s_0>$
- $S$ is a set of all states \{s_0, s_1, ..., s_l\}
- $I$ is a set of inputs \{i_0, i_1, ..., i_m\}
- $O$ is a set of outputs \{o_0, o_1, ..., o_n\}
- $F$ is a next-state function ($S \times I \rightarrow S$)
- $H$ is an output function ($S \rightarrow O$)
- $s_0$ is an initial state

Moore-type
- Associates outputs with states (as given above, $H$ maps $S \rightarrow O$)

Mealy-type
- Associates outputs with transitions ($H$ maps $S \times I \rightarrow O$)

Implicitly AND every transition condition with clock edge (FSM is synchronous)
An Elevator Controller FSM (Mealy)

\[ \begin{align*}
S &= \{ s_1, s_2, s_3 \} \\
I &= \{ r_1, r_2, r_3 \} \\
O &= \{ d_2, d_1, n, u_1, u_2 \} \\
f &: S \times I \rightarrow S \\
h &: S \times I \rightarrow O
\end{align*} \]
An Elevator Controller FSM (Moore)
Dataflow Graph Model (DFG)

- Most popular model for computation-intensive systems
- Its basic principles
  - Asynchrony: all operations execute when and only when all its operands are available
  - Functionality: operations are functions with no side-effects => execution order is not important
- Formal definition
  - \(<N,A,V,v^0,f>\)
Dataflow Graph Model (DFG)

- An FSM is a 5-tuple \( F \langle N, A, V, v_0, f \rangle \)
  - \( N \): is a set of all processing nodes \( \{ n_0, n_1, \ldots, n_l \} \)
  - \( A \): is a set of arcs between nodes \( \{ a_0, a_1, \ldots, a_m \} \)
  - \( V \): is a set of values associated with each arch \( \{ v_0, v_1, \ldots, v_n \} \)
  - \( V^0 \): is a set of initial values associated with each arch \( \{ v^0_0, \ldots, v^0_n \} \)
  - \( f \): is the function performed by each node
Dataflow Graph Model (DFG)

- Nodes represent transformations
  - May execute concurrently
- Edges represent flow of tokens (data) from one node to another
  - May or may not have token at any given time
- When all of node’s input edges have at least one token, node may fire
- When node fires, it consumes input tokens processes transformation and generates output token
FSM with Datapath Model (FSMD)

- Both *control* and *computation* are required in most systems
- FSMD is a combination of FSM and DFG models
- Formal definition
  - $<S,I,O,f,h,s_0>$
  - $f = \{f_C, f_D\}$
  - $h = \{h_C, h_D\}$
Model the elevator by only one state
- \textit{cfloor} and \textit{rfloor} variables
Neither FSM nor FSMD are suitable for complex systems
- No support for concurrency
- No support for hierarchy

Supporting Concurrency reduces system states

Supporting hierarchy made a model more Comprehensible
Hierarchical Concurrent FSM Model

- Extension of FSM, supporting concurrency and hierarchy
- Like FSM: sets of *states* and *transitions*
- Unlike FSM: each state can consist of *concurrent substates*
- An example language: Statecharts
What We Learned Today

- We use *Models* to decompose systems into pieces, then use *Languages* to generate a specification of those pieces.
  - VN, DE, FSM, DFG, FSMD, HCFSM, ...