System specification

- For every design, there exists a **conceptual view**

- Conceptual view depends on application
  - Computation: conceptualized as a program
  - Controller: conceptualized as a state-machine

- **Goal** of specification language
  - Capture conceptual view with minimum designer effort

- Ideal language
  - 1-to-1 mapping between conceptual model & language constructs
Outline

- Characteristics of commonly used conceptual models:
  Concurrency, hierarchy, synchronization

- Requirements for embedded system specification

- Evaluate HDLs with respect to embedded systems
  VHDL, Verilog, Esterel, CSP, Statecharts, SDL, SpecCharts
Concurrency

- **Behavior:** a chunk of system functionality
  - e.g. process, procedure, state-machine

- System often conceptualized as set of concurrent behaviors

- Concurrency can exist at different abstraction levels:
  - Job-level
  - Task-level
  - Statement-level
  - Operation-level
  - Bit-level

- Two types of concurrency within a behavior
  - Data-driven, Control-driven
Data-driven concurrency

- Operations execute when input data is available
- Execution order determined by data dependencies

1: \( Q = A + B \)
2: \( Y = X + P \)
3: \( P = (C - D) \times Q \)
Control-driven concurrency

- Control thread: set of operations executed sequentially
- Concurrency represented by multiple control threads

**Fork-join statement**

Sequential behavior $X$

```
begin
  Q();
  fork A(); B(); C(); join;
  R();
end behavior $X$;
```

**Process statement**

Concurrent behavior $X$

```
begin
  process A();
  process B();
  process C();
end behavior $X$;
```
State-transitions

- Systems often are state-based, e.g. controllers

- State may represent
  - mode or stage of being
  - computation

- Difficult to capture using programming constructs
Hierarchy

- Required for managing system complexity
  - Allows system modeler to focus on one subsystem at a time
  - Enhances comprehension of system functionality
  - Scoping mechanism for objects like types and variables

- Two types of hierarchy
  - Structural hierarchy
  - Behavioral hierarchy
Structural hierarchy

- System represented as set of interconnected components
- Interconnections between components represent wires
- Several levels: systems, chips, RT-components, gates
Behavioral hierarchy

- Ability to successively decompose behavior into sub-behaviors

- Concurrent decomposition
  - Fork-join
  - Process

- Sequential decomposition
  - Procedure
  - State-machine

behavior P
variable x, y;
begin
  Q(x);
  R(y);
end behavior P;
Programming constructs

- Some behaviors easily conceptualized as sequential algorithms

- Wide variety of constructs available
  Assignment, branching, iteration, subprograms, recursion, complex data types (records, lists)

```plaintext
type buffer_type is array (1 to 10) of integer;
variable buf : buffer_type;
variable i, j : integer;

for i = 1 to 10
  for j = i to i
    if (buf(i) > buf(j)) then
      SWAP(buf(i), buf(j));
    end if;
  end for;
end for;
```
Behavioral completion

- Behavior *completes* when all computations performed

- Advantages
  - Behavior can be viewed without inter-level transitions
  - Allows natural decomposition into sequential subbehaviors
Communication

- Concurrent behaviors exchange data

- Shared-memory model
  - Sender updates common medium
  - Persistent, Non-persistent

- Message-passing model
  - Data sent over abstract channels
  - Unidirectional / bidirectional
  - Point-to-point / multiway
  - Blocking / non-blocking

```
process P
begin
  variable x
  ....
  send (x);
  ....
end

process Q
begin
  variable y
  ....
  receive (y);
  ....
end
```
Synchronization

- Concurrent behaviors execute at different speeds

- Synchronization required when
  - Data exchanged between behaviors
  - Different activities must be performed simultaneously

- Two types of synchronization mechanisms
  - Control-dependent
  - Data-dependent
Control-dependent synchronization

- Synchronization based on control structure of behavior

Fork-join

```
behavior X
begin
  Q();
  fork A(); B(); C(); join;
  R();
end behavior X;
```

Reset

```
```

Diagram:

- Fork-join synchronization point
- Reset synchronization point

UC Irvine

Copyright (c) 1994 Daniel D. Gajski, Frank Vahid, Sanjiv Narayan, and Jie Gong
Data-dependent synchronization

- Synchronization based on communication of data between behaviors

Synchronization by common event

Synchronization by status detection

Synchronization by common variable
Exception handling

- Occurrence of event terminates current computation
- Control transferred to appropriate next mode
- Example of exceptions: interrupts, resets

![Diagram]

- P
- P1
- P2
- e
- Q
Timing

- Required to represent real world implementations

- **Functional timing**: affects simulation of system specification
  
  \[
  \text{wait for } 200 \text{ ns;} \\
  A \leq A + 1 \text{ after } 100 \text{ ns;}
  \]

- **Timing constraints**: guide synthesis and verification tools

\[
\begin{align*}
\text{behavior } B & \quad \text{max 10 ms} \\
\text{channel C} & \quad \text{(max 10 Mb/s)} \\
\text{behavior } Q & \\
\text{behavior } P
\end{align*}
\]
Embedded system specification

- Embedded system: behavior defined by interaction with environment

- Essential characteristics
  - State-transitions
  - Behavioral hierarchy
  - Programming constructs
  - Exceptions
  - Concurrency
  - Behavioral completion

![Diagram of embedded system specification]
VHDL

- IEEE standard, intended for documentation and exchange of designs [IEE88]

- **Characteristics supported**
  - Behavioral hierarchy: single level of processes
  - Structural hierarchy: nested blocks and component instantiations
  - Concurrency: task-level (process), statement-level (signal assignment)
  - Programming constructs
  - Communication: shared-memory using global signals
  - Synchronization: `wait on` and `wait until` statements
  - Timing: `wait for` statement, `after` clause in assignments

- **Characteristics not supported**
  - Exceptions: partially supported by guarded signal assignments
  - State transitions
Verilog and Esterel

- Verilog [TM91] developed as proprietary language for specification, simulation

- Esterel [Hal93] developed for specification of reactive systems

- **Characteristics supported:**
  - Behavioral hierarchy: fork-join
  - Structural hierarchy: hierarchy of interconnected *modules*
  - Programming constructs
  - Communication: shared registers (Verilog) and broadcasting (Esterel)
  - Synchronization: *wait* for an event on a signal
  - Timing: modeling of gate, net, assignment delays in Verilog
  - Exceptions: *disable* (Verilog), *watching*, *do-upto*, *trap* statements (Esterel)

- **Characteristics not supported:** State transitions
**SDL (Specification and Description Language)**

- CCITT standard in telecommunication for protocol specification [BHS91]

**Characteristics supported**
- Behavioral hierarchy: nested data
- Structural hierarchy: nested blocks
- State transitions: state machine in processes
- Communication: message passing
- Timing: *timeouts* generated by timer object

**Characteristics not supported**
- Exceptions
- Programming constructs
CSP (Communicating Sequential Processes)

- Intended to specify programs running on multiprocessor machines [Hoa78]

- **Characteristics supported**
  - Behavioral hierarchy: fork-join using parallel command
  - Programming constructs
  - Communication: message passing using input, output commands
  - Synchronization: blocking message passing

- **Characteristics not supported**
  - Exceptions
  - State transitions
  - Structural hierarchy
  - Timing
SpecCharts

- Developed for embedded system specification [NVG92]
- PSM (program-state machine) model + VHDL

**Characteristics supported**
- Behavioral hierarchy: sequential/concurrent behaviors
- State transitions: TOC (transition on completion) arcs
- Communication: shared memory, message passing
- Exceptions: TI (transition immediately) arcs

**Characteristics similar to VHDL**
- Programming constructs
- Structural hierarchy
- Synchronization and Timing
SpecCharts : state transitions

- State transitions represented by TOC and TI arcs between behaviors

```
behavior MAIN type sequential subbehaviors is
begin
  P : (TOC, u, Q);
  Q : (TOC, v, P), (TOC, w, R);
  R : (TOC, x, Q);

  behavior P ....
  behavior Q ....
  behavior R ....
end MAIN;
```
SpecCharts: behavioral hierarchy

- Hierarchy represented by nested behaviors
- Behavior decomposed into sequential or concurrent subbehaviors

```
behavior MAIN  type sequential subbehaviors is
begin
  P : (TOC, true, Q_R);
  Q_R : (TOC, true, S);
  S : ;
end P ....

behavior Q_R type concurrent subbehavior is
begin
  Q : (TOC, true, halt);
  R : (TOC, true, halt);

  behavior Q ....
  behavior R ....
end Q_R;

behavior S ....
end MAIN;
```
SpecCharts : exceptions

- Exceptions represented by TI (transition immediately) arcs

```
behavior MAIN type sequential subbehaviors is
begin
  P : (TI, e, Q);
  Q : ;
  behavior P
  behavior P1
  behavior P2
  Q : ;
end MAIN;
```
## Summary

<table>
<thead>
<tr>
<th>Language</th>
<th>Embedded System Features</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>State Transitions</td>
</tr>
<tr>
<td>VHDL</td>
<td>○</td>
</tr>
<tr>
<td>Verilog</td>
<td>○</td>
</tr>
<tr>
<td>Esterel</td>
<td>○</td>
</tr>
<tr>
<td>SDL</td>
<td>●</td>
</tr>
<tr>
<td>CSP</td>
<td>○</td>
</tr>
<tr>
<td>Statecharts</td>
<td>●</td>
</tr>
<tr>
<td>SpecCharts</td>
<td>●</td>
</tr>
</tbody>
</table>

- **○** Feature fully supported
- **○○** Feature partially supported
- **●** Feature not supported