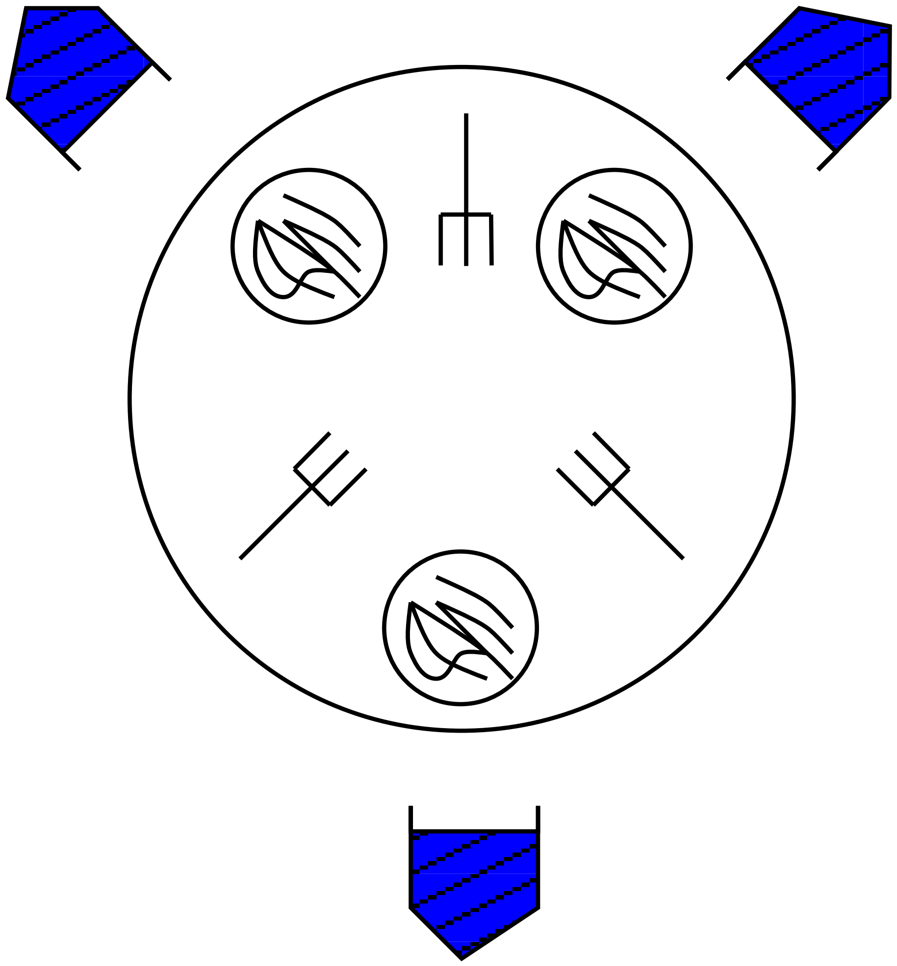
2.4.5 Predicate/Transition nets

Goal: compact representation of complex systems:

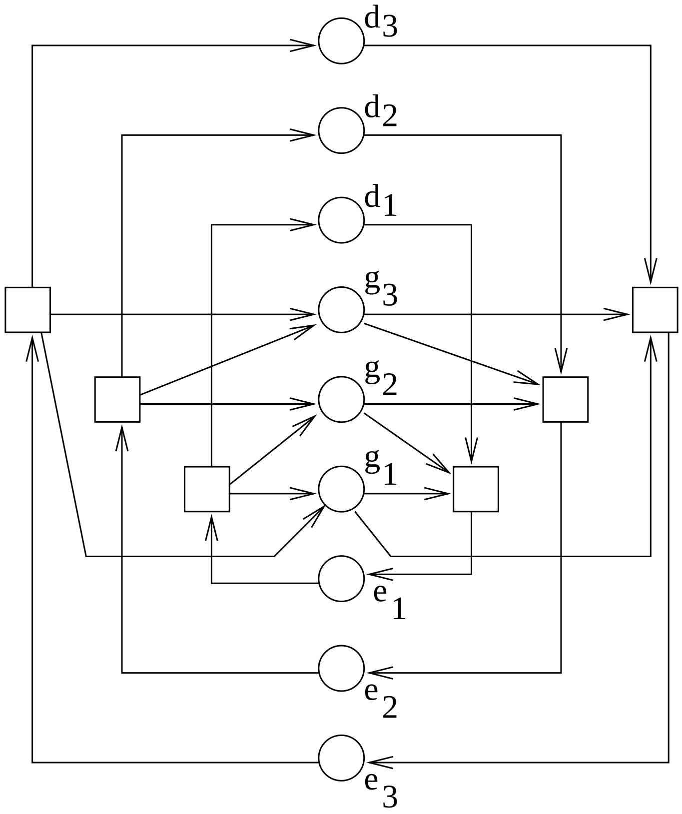
Example: *Dining philosophers*.



Boolean variables

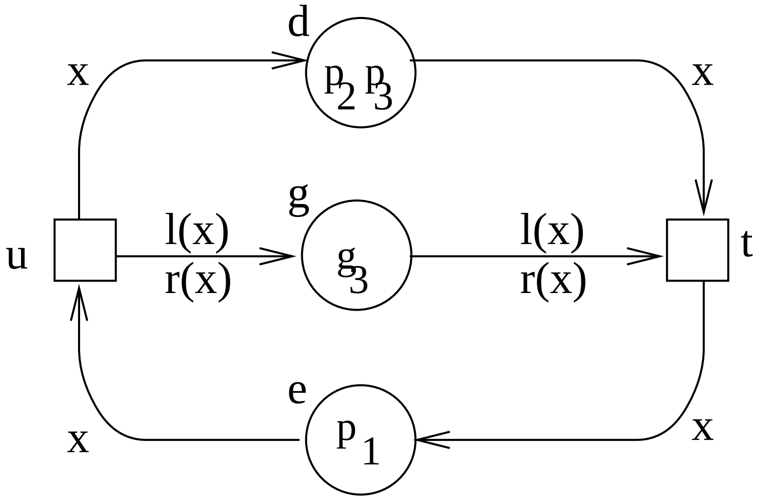
d_i: i thinking, e_i: i eating, g_i: spoon i available.

Condition/event net:



Predicate/Transition net:

$l(x)$ and $r(x)$, left resp. right spoon of x.



Properties:

Tokens: individuals

Modell can be extended to arbitrary number of philosophers.

Semantics can be defined by replacing by the equivalent condition/event net.

2.4.6 Evaluation

Appropriate for distributed applications.

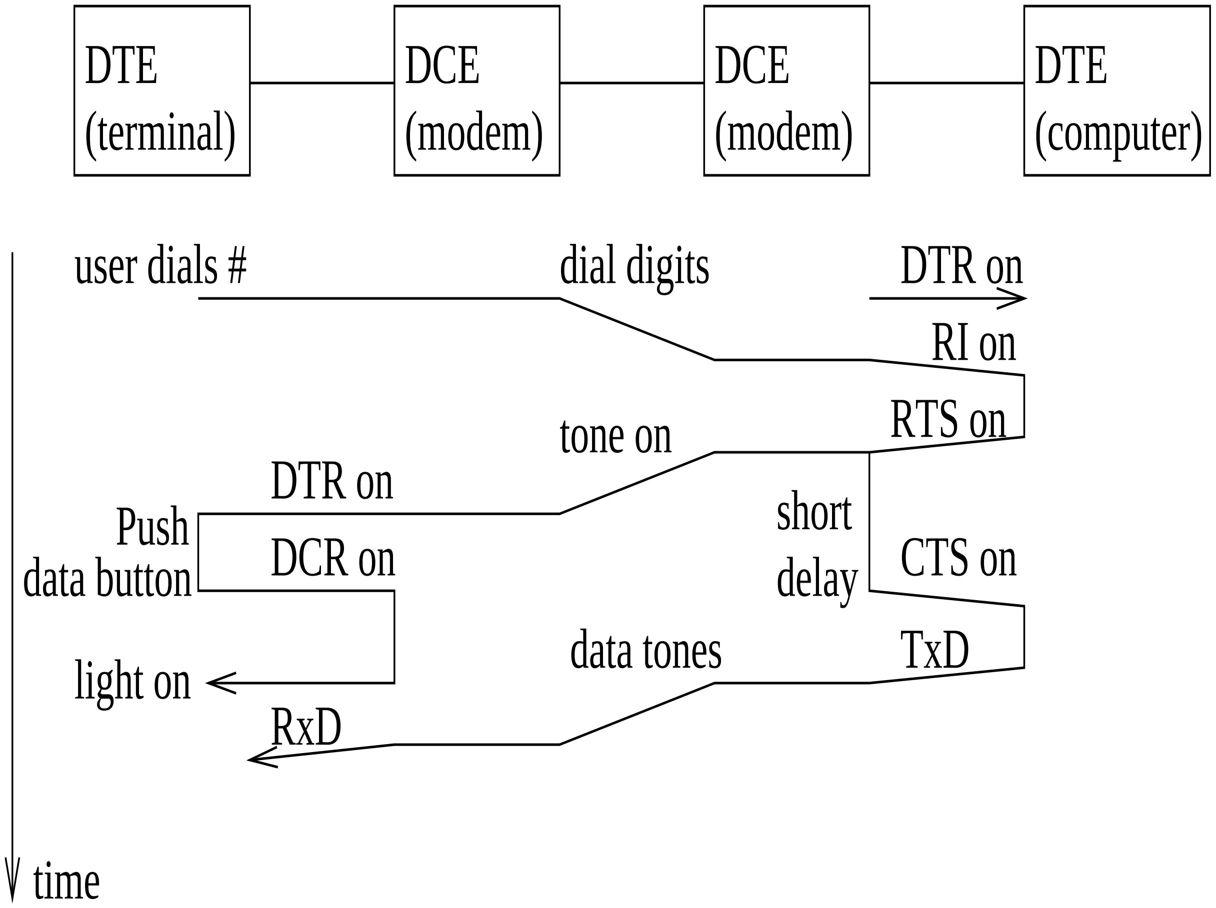
Well-known theory for formally proving properties.

For nets presented so far:
- no programming elements,
- no timing information,
- no hierarchy.

2.5 Message sequencing charts (MSCs)

Used for representing acitivities within distributed systems.

Example: Startup phase of an RS 232 connection:



Also used for representing train schedules.

MSCs only show examples.

Do not show tolerances and constraints.

Not useful for formal techniques.

2.6 UML

UML (Unified Modelling Language)

[Oesterreich97, Fowler98]

Language for modelling object-oriented software development.

UML includes:

1. **State diagrams**:
   UML includes StateCharts.
2. **activity diagrams**:
   activity diagrams = extended Petri nets.
   Extensions: symbols for decisions.
   SDL-like placement
3. **interaction diagrams**:
   Equivalent to message sequencing charts.
4. **class diagrams**:
   Represent inheritance of classes.
5. **package diagrams**:
   correspond to module charts of StateMate.
6. **use cases**: describe typical applications.

Checking of consistency?

2.7 ADA

Language designed for DoD.

Current version: ADA 1995

(object oriented extension of original language)

Salient feature:

Rendez-vous model of communication

(Processes have to meet to exchange information):

Example:

```
TASK screen_output IS
 ENTRY call_ch(val:character; x, y: integer);
 ENTRY call_int(z, x, y: integer);
END screen_out;
TASK BODY  screen_output IS
...
 SELECT
  ACCEPT call_ch ...  DO ..
  END call_ch;
 OR
  ACCEPT call_int ... DO ..
  END call_int;
 END SELECT;
...
```

Sending a message:

```
BEGIN
 screen_out.call('Z',10,20);
 EXCEPTION
  WHEN tasking_error => (exception handling)
END;
```

2.8 Java

Initially designed for embedded systems.

Advantages of using Java for embedded systems:

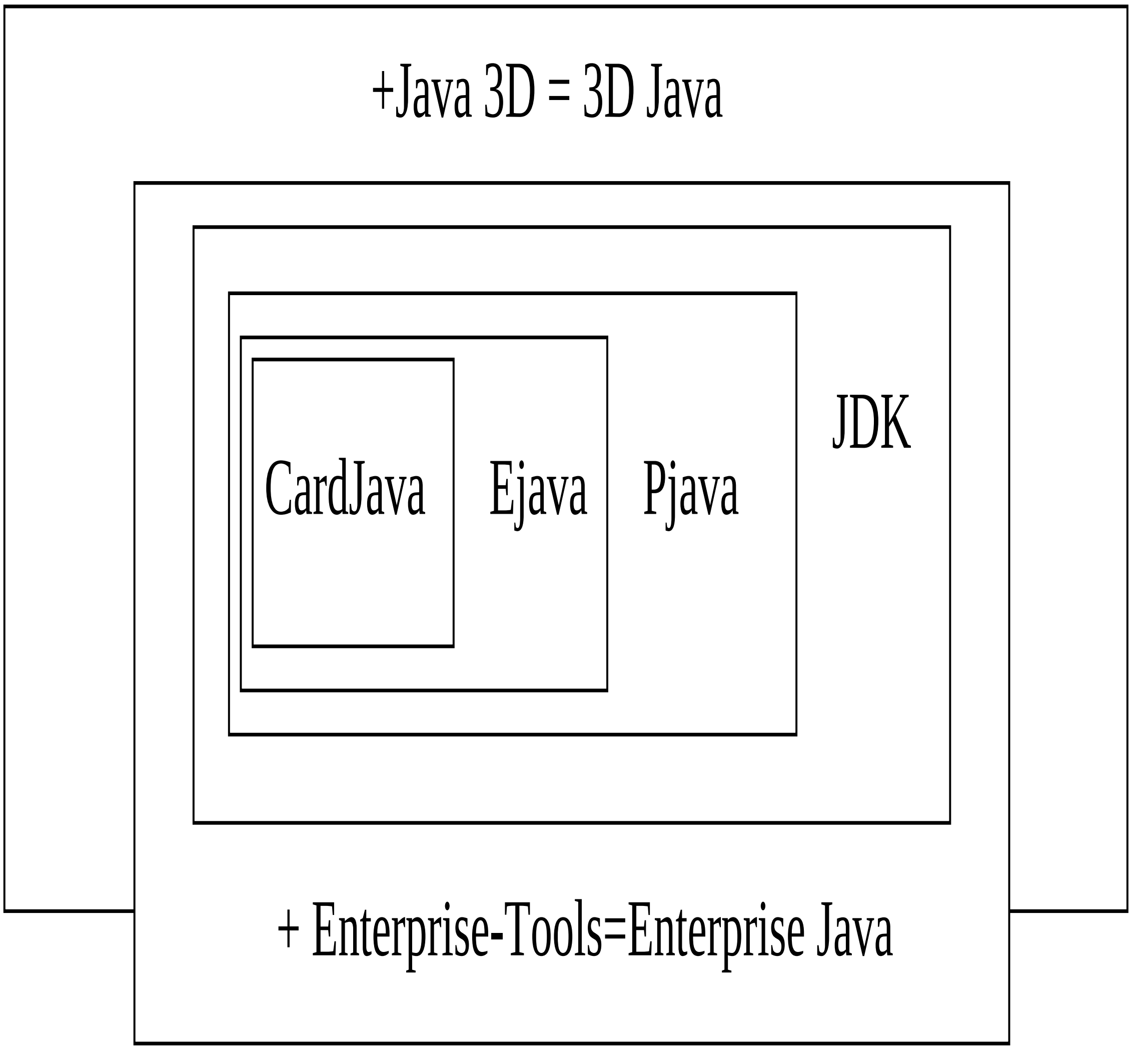
- **object orientation**
- **supports multiple threads**
- Java is **simple**.
- **short design cycles**
- Java is **compact**. Basic interpreter: 40 kBytes, Thread-Support, libraries: additional 175 kBytes.
- Java is **robust**: No memory leaks.
- Java is **plattform independent**
- **dynamic loading of classes**.
- Java is **safe**: security checks etc.

Problems:

- **automatic garbage collection**
- **Unspecified dispatcher**

Working groups are trying to solve the problems (Nilsen and others).

Different subsets discussed



Considered for set top boxes,
Mercedes-Benz cars etc.

2.9 VHDL

VHDL = VHSIC Hardware Description Language

VHSIC = very high speed integrated circuit

Definition started by DoD in 1980.

1984: first version of the language defined.

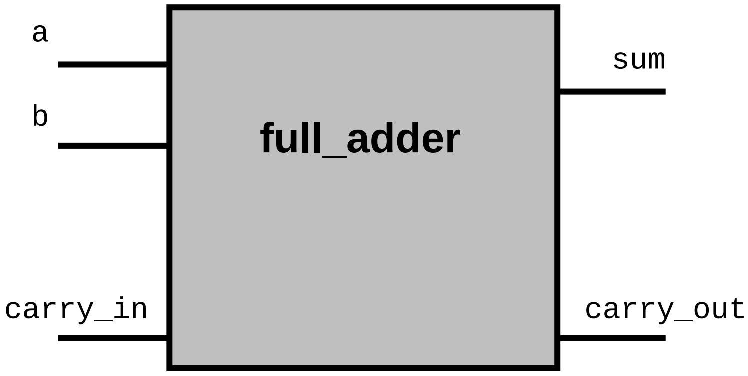
1987: revised version became IEEE Standard 1076.

1992: revised IEEE Standard.

Salient features:

- Syntax based on ADA
- flexible definition of value sets
- supports bitvectors
- supports time and time units
- processes are static, non-terminating and flat
- 2-phase simulation semantics to avoid non-determinism
- includes configuration mechanism
- Parallelism of hardware modelled by processes.
- Several processes per hardware component possible.

Example: full adder



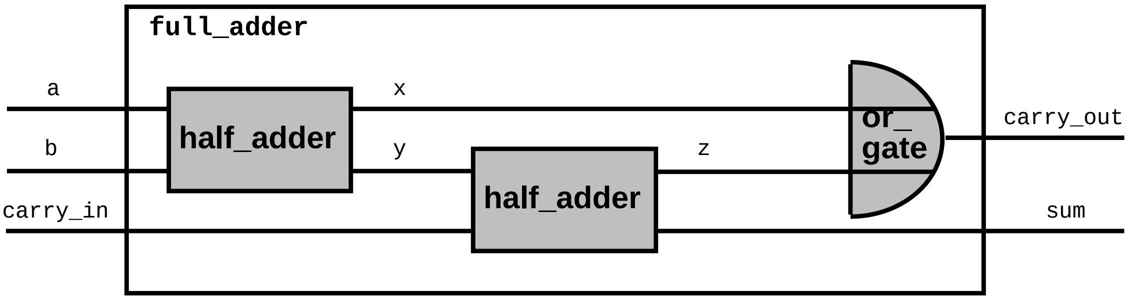
Header:

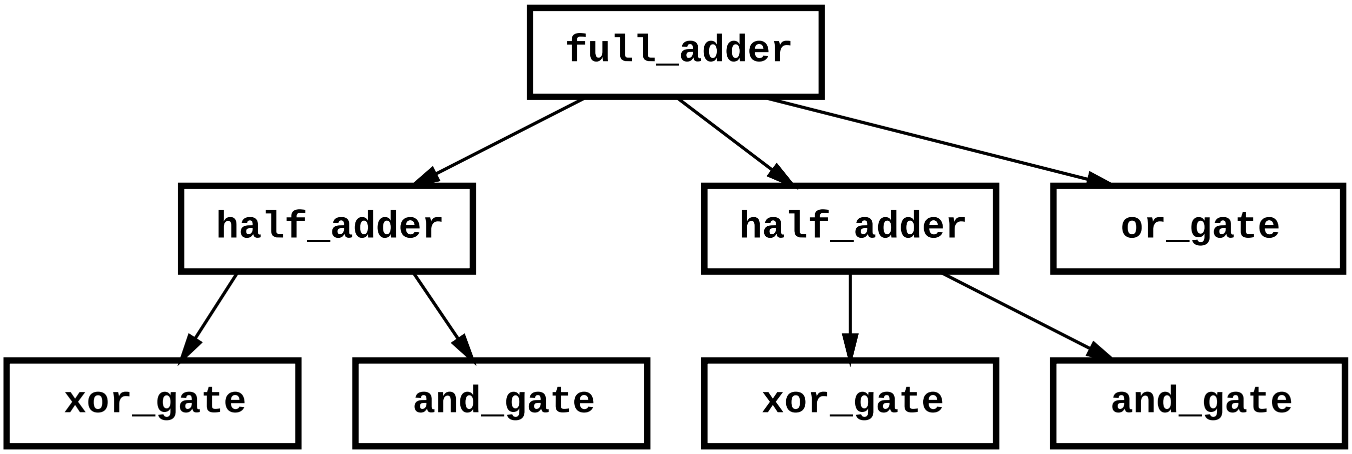
```
ENTITY full_adder IS
  PORT (a, b, carry_in: IN Bit;
        sum, carry_out: OUT Bit);
END full_adder;
```

Behavioural body:

```
ARCHITECTURE behavior OF full_adder IS
 SIGNAL s: Bit;
 BEGIN
  s        <= a xor  b AFTER 5 Ns;
  sum      <= s xor carry_in AFTER 5 Ns;
  carry_out<= (a and b) or
             (s and carry_in) after 10 Ns;
 END behavior;
```

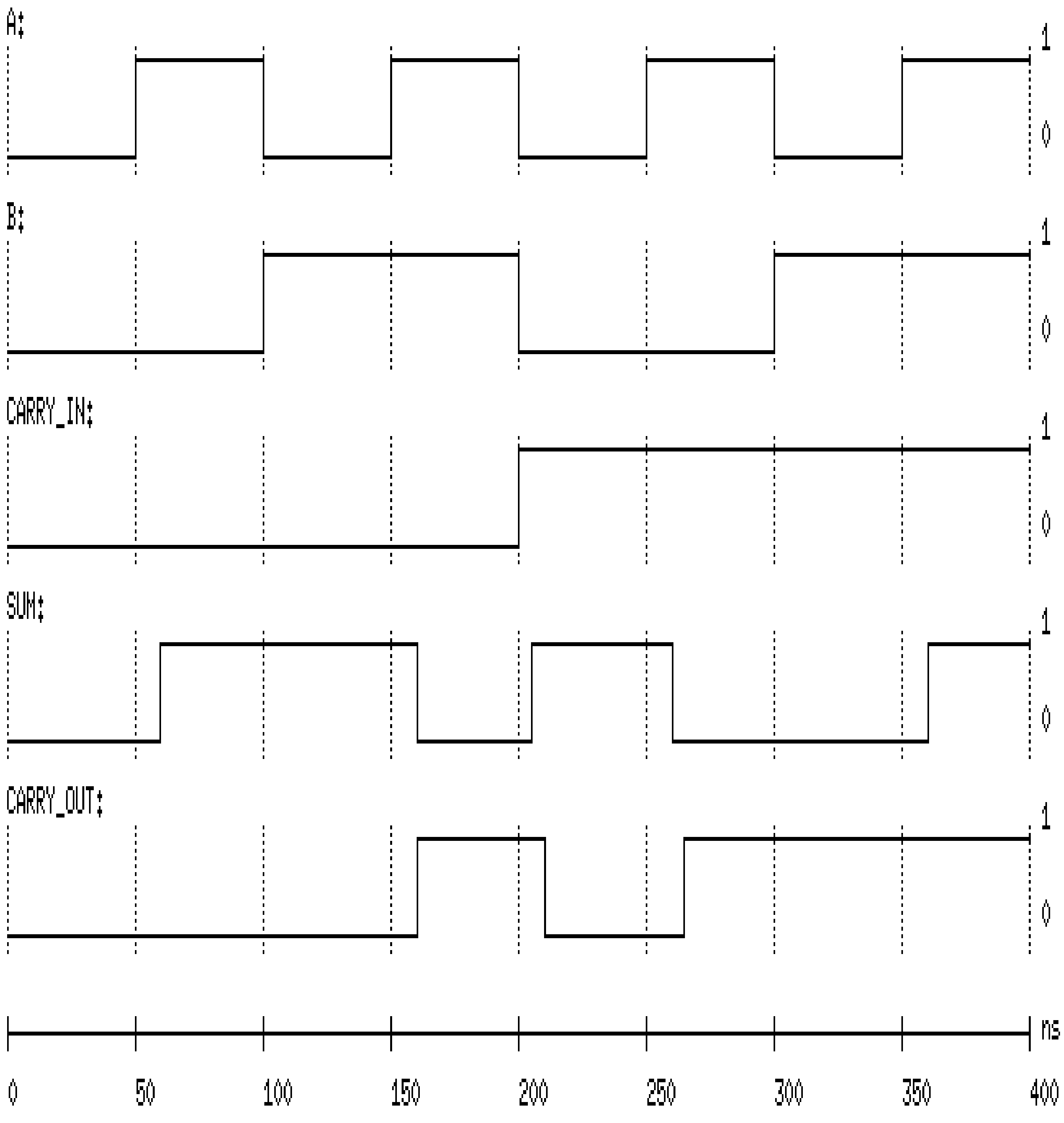
Structural body:






```
ARCHITECTURE structure OF full_adder IS
  COMPONENT half_adder
    PORT (i1, i2:IN Bit; carry:OUT Bit; sum:OUT Bit);
  END component;
  COMPONENT or_gate
    PORT (i1, i2:IN Bit; o:OUT Bit);
  END component;
  SIGNAL x, y, z: Bit;
 BEGIN
  s1: half_adder PORT MAP (a, b, x, y);
  s2: half_adder PORT MAP (y, carry_in, z, sum);
  s3: or_gate    PORT MAP (x, z, carry_out);
 END structure;
```

Simulation results:

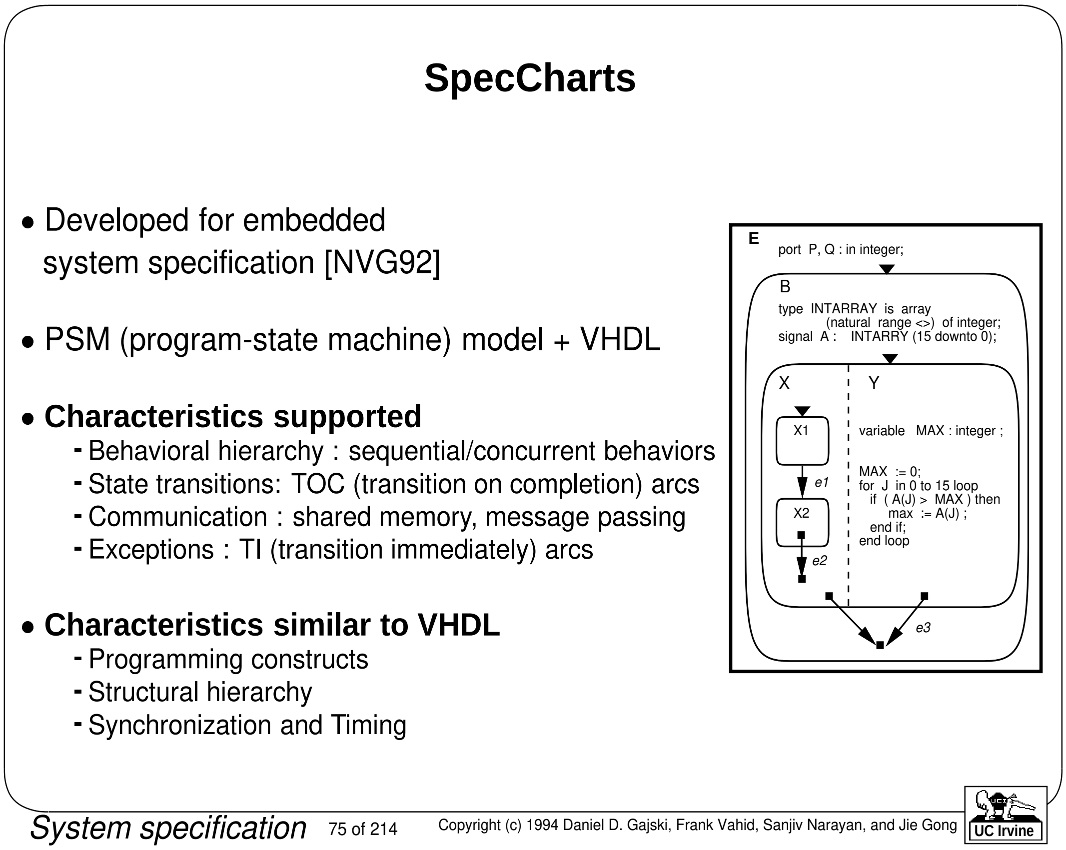


2.10 Other languages

1. **SpecCharts**

   SpecCharts [Gajski94] is a StateChart variant using VHDL to denote computations.

   Characteristics of StateCharts:



   Two types of transitions: *transition on completion* and *transition immediately*.

   SpecCharts descriptions can be easily translated into VHDL.

2. **Estelle**

Scope: description of communication protocolls.

Standard defined by ISO.

Scope similar to SDL.

Attempts to integrate Estelle with SDL failed.

Communication based on channels +
FIFO buffers.

FSM-based modell of components.

Textual representation only. Example:

```
state disconnected,suspended,connected;
trans
{1} from disconnected to suspended
      when User.ICONreq
      begin output PDU.CR end;
{2} from suspended to connected
      when PDU.CC
      begin output User.ICONconf end;
{3} from suspended to disconnected
      when PDU.DR
      begin output User.IDISind end;
{4} from disconnected to connected ...
```

3. **CSP** [Hoare]

CSP = communicating sequential processes

One of the first languages containg language elements for describing interprocess communication.

Communication based on named channels.

Example:

```
PROCESS A             PROCESS B
.....                 ......
VAR a ..              VAR b ..
 a := 3;              ...
 c!a   ;               c?b   -- input from channnel c
END;                  END;
```

Processes waiting for meeting at rendez-vous point.

Blocking communication.

CSP formed the basis for OCCAM (language for transputers [INMOS]).

Earlier transputers: 4 hardware communication channels

Within a single transputer, communication is handled by the operating system.

More recent transputers: virtual channels. Communic. with any number of processors.

4. **Verilog**

   Another hardware description language.

   Less flexible than VHDL, but usually simulates faster.

   Very popular in the US.

5. **Z**

   language for algebraic specification.

6. **LOTOS**

   Another language for algebraic specification.

7. **Esterel**

   Yet another language for algebraic specification.

8. **Silage**

   Special functional language for signal processing.

9. **HardwareC**

   Hardware description language with syntax similar to C.

   Simple extensions for describing concurrent and parallel execution and bitvectors.

2.11 Comparison of languages
Comparison according to Gajski:

Summary

Language	Embedded System Features					
	State Transitions	Behavioral Hierarchy	Concurrency	Program Constructs	Exceptions	Behavioral Completion
VHDL	○	◐	●	●	○	●
Verilog	○	●	●	●	●	●
Esterel	○	●	●	●	●	●
SDL	●	◐	●	○	○	●
CSP	○	●	●	●	○	●
Statecharts	●	●	●	○	●	○
SpecCharts	●	●	●	●	●	●

● Feature fully supported ◐ Feature partially supported ○ Feature not supported

System specification 79 of 214 UC Irvine

Remarks:

- Esterel can be used to describe state transitions
- SDL contains programming language elements.
- contains only criteria for which SpecCharts looks good (e.g. structural hierarchy not included, no dynamic process creation considered)

Comparison according to Niemann:

	Standardization	Availability of Tools	Model Executability	Formal Analysis	Non-determinism	Timing Aspects	Exceptions	Synchron./Communication	Concurrency	Behavioural Completion	State-Transitions	Programming Constructs	Behavioural Hierarchy	Structural Hierarchy
Lotos	+	0	-	+	+	-	+	+	+	+	+	+	+	+
SDL	+	0	+	+	+	0	-	+	+	+	+	-	0	+3
Esterel	-	0	+	+	-	-	+	+	+	+	-	+	+	+
StateCharts	-	0	+	-1	-	0	+	+	+	-	+	-	+	-
High-Level Petri Net	-	0	+	+	-4	+6	+	+	+	-	+	-	-	+5
VHDL	+	+	+	-2	-	+	-	+	+	+	-	+	0	+
Verilog	-7	+	+	-	-	+	+	+	+	+	-	+	+	+
SpecCharts	-	0	+	-	-	+	+	+	+	+	+	+	+	-
HardwareC	-	0	+	-	-	0	-	+	+	+	-	+	0	+
CSP/Occam	-	0	+	+	-	-	-	+	+	+	-	+	+	-
Silage	-	0	0	-	-	0	-	-	+	+	-	-	-	-

Remarks:

1. Formal semantics of StateCharts now available.
2. Possible formal semantics of VHDL described in book by Delgado-Kloos (ed.) .
3. SDL supports only mix of structural and behavioural hierarchy.
4. Non-determinism included in Petri net model.
5. Same as for SDL.
6. There are timed Petri nets. Formal properties?
   Insufficient knowledge about variants of Petri nets.
7. IEEE standard for Verilog exists.

→ no single language that meets all requirements.
Mix of languages used in practice.

Translation between languages proposed by several researchers.