The hardware software co-design problem is posed as an evolution of existing synthesis Methods.
**Vulcan**

1. Specification
2. Modeling
3. Constraint Analysis
4. Software and Runtime Environment
5. Target Architecture - H/S Interface
6. Partitioning
7. Co-simulation

---

**Model -- Flow Graph**

- Hierarchical control/data-flow graph
  - control flow primitives (iteration and model call)
  - modeled through hierarchy
- Acyclic
  - models partial order of tasks/operations
  - iteration is modeled outside the graph
- Polar
  - source and sink vertices model No-Operations

---

Margarida Jacome - UT Austin - Spring 98
Flow Graph (CDFG)

- **Conjoined output**: directs the flow of control to all its branches.
- **Disjoined output**: selects one of the successors based on a condition index.

---

**Operation vertices in a Flow Graph**

1. **no-op**: no operation
2. **cond**: conditional fork
3. **join**: conditional join
4. **op-logic**: logical operations
5. **op-arithmetic**: arithmetic operations
6. **op-relational**: relational operations
7. **op-io**: I/O operations
8. **wait**: wait on a signal variable
9. **link**: hierarchical operations
   - **call**: procedure call (invocation times = 1)
   - **loop**: iteration (invocation times ≥ 1)

---

Margarida Jacome - UT Austin - Spring 98
**System Model**

Consists of one or more flow graphs that may be hierarchically linked to other flow graphs

System Model: \( \Phi = \{G_1^*, G_2^*, ..., G_n^*\} \)

where

\[ G_i^* \text{ represents the process graph model } G_i \text{ and all the flow graphs that are hierarchically linked to } G_i. \]

A flow graph model that is common to two hierarchies of a system model is called a *shared model*.

---

**Flow Graphs: Execution Semantics**

1. At any time, an operation may be
   - waiting for execution
   - presently executing
   - having completed its execution

   The state of a vertex is defined as being one of \( \{s_r, s_e, s_d\} \)

\[ s_r: \text{ reset state} \rightarrow \text{waiting for execution} \]

\[ s_e: \text{ enable state} \rightarrow \text{presently executing} \]

\[ s_d: \text{ done state} \rightarrow \text{completed execution} \]
### Example

<p>| | | | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- ➡️ reset
- ➡️ enable
- ➡️ done

No assumption about timing of the operations is made ➞
(consecutive rows can be spaced arbitrarily over the time axis)

### Timing Properties

1. Operation delay
2. Graph Latency
3. Rate of Execution (operations)
4. Rate of Reaction (graphs)
**Operation Delay**

\[ \delta(v_f) = 1 \]

**Latency**

Latency, \( \lambda(G) \), of a graph model \( G \) refers to the execution delay of \( G \)

\[ \lambda_k(G) = t_k(v_n) - t_k(v_0) \]

- the latency of a non-hierarchical flow graph may be variable due to the presence of conditional paths
Execution Delay of Link Vertices

Given by

- latency of the corresponding graph model times the number of times the called graph is invoked
- execution delay of a link vertex can be
  - variable
  - unbounded (loop vertices with unbounded indices)

Link vertices: call and/or loop (point to other flow graphs in the hierarchy)

Timing Properties

1. Operation delay
2. Graph Latency
3. Rate of Execution (operations)
4. Rate of Reaction (graphs)
### Rate of Execution (operations)

1. Assuming a synchronous execution model with cycle time $\tau$, the rate of execution at invocation $k$ of operation $v_i$ is given by the time interval between its current and previous execution

$$\rho_i(k) := \frac{1}{t_k(v_i) - t_{k-1}(v_i)} \quad \text{(sec$^{-1}$)}$$

$$= \frac{\tau}{t_k(v_i) - t_{k-1}(v_i)} \quad \text{(cycle$^{-1}$)}$$

### Rate of Reaction (Graphs)

1. For a graph model, $G$, its rate of reaction is defined as the rate of execution of its source operation

$$\rho_G(k) := \rho_0(k)$$

The reaction rate is used to capture the effect on the run-time system of the type of implementation chosen for the graph model.

- e.g., the choice of a non-pipelined implementation leads to

$$\rho_G(k) - \rho_G(k-1) = \lambda_k(G) + \gamma_k(G)$$

where $\gamma_k(G)$ represents the overhead delay (delay of reinvocation of $G$).
**Timing Properties**

1. Operation delay
2. Graph Latency
3. Rate of Execution (operations)
4. Rate of Reaction (graphs)

Fixed, variable, bounded/unbounded

**Non-Determinism**

1. The delay of an operation may be variable, depending on:
   - The value of input data: e.g., loops with data dependent iteration counts, call vertices with conditionals
   - The timing of input data: e.g., wait operation
2. The latency of a graph may be variable
**Data Dependent Delays**

- **1**: Data-dependent loop and synchronization operations introduce uncertainty over
  - the precise execution delay of the model
  - the order of execution of the operations in the model

Operations with **variable delays** are termed **non-deterministic delay** or **ND** operations.

---

**Non-determinism and Execution Rate**

- **1**: Data-dependent loop and synchronization operations introduce uncertainty over
  - the precise execution delay of the model
  - the order of execution of the operations in the model

Operations with **variable delays** are termed **non-deterministic delay** or **ND** operations.
**A Single Rate Model**

On each execution of the flow graph, each operation executes once.

In this case, the reaction rate of the graph $G$ is:

$$\rho_G(k) := \rho_0(k) = \rho_{vi}(k),$$

for all $vi \in V(G)$ and for all $k \geq 0$.

The execution of $G$ proceeds at **single rate**.

---

**A Multi Rate Model**

---

Page 11
Timing Constraints and Constraint Analysis

1 Timing Constraints
2 Scheduling
3 Constraint Satisfiability

Timing Constraints

1 Operation delay constraints
   - *unary*: bounds on the delay of an operation
   - *binary*: bounds on the delay between the starting time of two operations

1 Execution rate constraints
Binary Delay Constraints

**Minimum timing constraint**, \( l_{ij} \geq 0 \) from operation vertex \( v_i \) to \( v_j \) is defined as

\[
t_k(v_j) \geq t_k(v_i) + l_{ij}
\]

by default, any sequencing dependency between two operations induces a minimum timing constraint

**Maximum timing constraint**, \( u_{ij} \geq 0 \) from operation vertex \( v_i \) to \( v_j \) is defined as

\[
t_k(v_j) \leq t_k(v_i) + u_{ij}
\]

Example

```
MAX DELAY 3

MIN DELAY 4
```

```
1  NOP
2  1
3  2
4  3
```

```
0 + 1
2 + 3
```

```
0 + 1
2 + 3
```

```
0 + 1
2 + 3
```

```
0 + 1
2 + 3
```

```
0 + 1
2 + 3
```

```
0 + 1
2 + 3
```

```
0 + 1
2 + 3
```

```
0 + 1
2 + 3
```

```
0 + 1
2 + 3
```

```
0 + 1
2 + 3
```

```
0 + 1
2 + 3
```

```
0 + 1
2 + 3
```

```
0 + 1
2 + 3
```

```
0 + 1
2 + 3
```

```
0 + 1
2 + 3
```

```
0 + 1
2 + 3
```

```
0 + 1
2 + 3
```

```
0 + 1
2 + 3
```

```
0 + 1
2 + 3
```
### Operation Delay Constraints

Can capture durational and deadline constraints in specifying real time systems.

- **a before b**: $a \delta_a b$
- **a meets b**: $a \ldots b$
- **a overlaps b**: $a \ldots b$
- **a finishes by b**: $a \delta_a b$
- **a during b**: $a \delta_a 0$
- **a after b**: $a \delta_b b$

etc...

### Timing Constraints

1. **Operation delay constraints**
   - refer to constraints on the interval of time between successive executions of an operation

2. **Execution rate constraints**
   - rate constraints on input (output) operations refer to the rates at which the data is required to be produced (consumed)
Data Rate Constraints

1 **Minimum data rate constraint**, \( r_i \) (cycles\(^{-1}\)), on an input/output operation \( v_i \) defines a lower bound on the execution rate of the operation

\[
\rho_{v_i}(k) \geq r_i \quad \forall k > 0 \quad [\text{min rate}]
\]

\( \implies t_{k}(v_i) - t_{k-1}(v_i) \leq \tau \cdot r_i^{-1} \quad \forall k > 0 \)

1 **Maximum data rate constraint**, \( R_i \) (cycles\(^{-1}\)), on an I/O operation \( v_i \) defines an upper bound on the execution rate of the operation

\[
\rho_{v_i}(k) \leq R_i \quad \forall k > 0 \quad [\text{max rate}]
\]

\( \implies t_{k}(v_i) - t_{k-1}(v_i) \geq \tau \cdot R_i^{-1} \quad \forall k > 0 \)

---

### Ex.: Specification of Data Rate Constraints

```plaintext
process example (a,b,c)
    in port a[8], b[8];
    out port c[8];
    {
        boolean x[8], y[8], z[8], w[8];
        tag A;
        x = read(a);
        y = read(b);
        z = x * y;
        w = x + y;
        while(z >= 0) {
            while(w >= 0) {
                write c = y;
                w = w - 1;
            }
            z = z - w;
            write c = z;
        }
    }
```

**A:**
- attribute "constraint minrate of A = 100 cycles/sample"
- attribute "constraint minrate 0 of A = 1 cycles/sample"
- attribute "constraint minrate 1 of A = 10 cycles/sample"

---

**Note:** The image contains a diagram illustrating the relative min constraints indexed by the corresponding loops. The diagram shows a graph with nodes labeled \( A \) and \( G \), and an edge \( v_i \) connecting them. The relative min rate constraint is indicated as 0.01 per cycle.
Timing Constraints

Scheduling

Constraint Satisfiability

For each invocation of a flow graph model, an operation is invoked zero, one, or many times depending upon its position on the hierarchy of the flow model.

The execution times $t_k(v)$ of an operation $v$ are determined by two separate mechanisms:

- The runtime scheduler, $\gamma$ determines the invocation time of flow graphs.
- The operation scheduler, $\Omega$. 

Diagram: Flow graph with nodes $G$, $G_1$, $v_1$, $v_2$, and edge $x$.
### Timing Constraints and Scheduling

Given a scheduling function, a timing constraint is considered satisfied if

- the operation starting times determined by the scheduling function satisfy the inequalities

\[
\begin{align*}
    t_k(v_j) &\geq t_k(v_i) + l_{ij} & \text{[min delay]} \\
t_k(v_j) &\leq t_k(v_i) + u_{ij} & \text{[max delay]} \\
\rho_{cv}(k) &\leq R_i & \text{[max rate]} \\
\rho_{cv}(k) &\geq r_i & \text{[min rate]}
\end{align*}
\]

### Relative Scheduler

For a given vertex \( v_i \), a set \( A(v_i) \) of anchor vertices is defined as the set of conditional (CD) and loop, wait (ND) vertices that have a path to \( v_i \).

\[
A(v_i) = \{ v_j \in V : v_j \succ v_i, \ v_j \text{ is ND or CD} \}
\]

A relative schedule function \( \Omega_i \) is defined as a set of offsets for each operation such that the operation start time satisfies

\[
t_k(v_j) \geq \max_{a \in A(v_i)} \left[ t_k(a) + \delta(a) + \theta_a(v_i) \right]
\]
### Constraint Graph

Anchors??

### Modified Relative Schedule

<table>
<thead>
<tr>
<th>Vertex</th>
<th>$v_1$</th>
<th>$v_3$</th>
<th>$v_8$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$v_1$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$v_2$</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$v_3$</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$v_4$</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>$v_5$</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>$v_6$</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>$v_7$</td>
<td>(2,4)</td>
<td>(1,3)</td>
<td></td>
</tr>
<tr>
<td>$v_8$</td>
<td>(2,4)</td>
<td>(1,3)</td>
<td></td>
</tr>
<tr>
<td>$v_9$</td>
<td>--</td>
<td>--</td>
<td>--</td>
</tr>
<tr>
<td>$v_{10}$</td>
<td>--</td>
<td>--</td>
<td>--</td>
</tr>
<tr>
<td>$v_{11}$</td>
<td>--</td>
<td>--</td>
<td>--</td>
</tr>
<tr>
<td>$v_{12}$</td>
<td>--</td>
<td>--</td>
<td>--</td>
</tr>
<tr>
<td>$v_{13}$</td>
<td>--</td>
<td>--</td>
<td>0</td>
</tr>
</tbody>
</table>
**Modified Relative Schedule**

**Constraint Satisfiability**

For constraint analysis purposes, it is not necessary to determine a schedule of the operations, but only to verify the existence of a schedule identifying conditions under which no solution (i.e., schedule) exists.
**ND operations**

In the presence of *ND* operations

- satisfiability analysis attempts to determine the existence of a schedule of operations for all possible (and conceivable) values of the delay of the *ND* operations

**Modified relative scheduling - Min/Max Delay Constraints**

- A minimum delay constraint is always satisfiable
  
  \[ t_k(v_i) \geq \max_{a \in A_a(v_i)} [ t_a + \delta(a) + |q_a(v_i)| \infty] \]

  For each constraint \( lij \) solution can be constructed such that
  
  \[ \theta_{ij}(v_i) \geq \max (l(i(v_i)v_j), l_{ij}) \]

- A maximum delay constraint may not always be satisfiable

**Satisfiability - Delay Constraints**

**Feasibility:**

- A constraint graph is considered *feasible* if it contains no positive cycle when the delay of the ND operations is assigned to zero.

**Condition necessary and sufficient** to determine the satisfiability of constraints in the presence of ND operations:

- Operation delay constraints are *satisfiable* if and only if
  
  the constraint graph is *feasible*
  
  there exists no cycles with ND operations
Examples

Constraints are not satisfiable (maybe feasible)

Can be modified such that...

Constraints are satisfiable

---

Max Rate Constraints

A max-rate constraint, $R_i$, in $G$ is satisfied if

$$l_m(G) \geq R_i^{-1}$$

As with minimum delay constraints, maximum rate constraints are always satisfiable when the lower bound $l_m(G) \leq R_i^{-1}$, the max-rate constraint can still be satisfied by an appropriate choice of overhead delay that is applicable to every execution of $G$. 
**Min Rate Constraint**

A minimum rate constraint \( r_i \) on an operation \( v_i \in V(G) \), where \( G \) contains no ND operations is satisfiable if

\[
\overline{\gamma}(G) + l_M(G) \leq \left( \frac{t}{r_i} \right)
\]

1. A minimum rate constraint places an upper bound on the interval of successive executions of an operation.

**Min Rate Constraints**

General case: involves two bounds

\[
\overline{\gamma}(G) + \overline{\lambda}(G) \leq \left( \frac{t}{r_i} \right)
\]
**Min Rate Constraints**

General case: involves two bounds

\[ \tilde{\gamma}(G) + \tilde{\lambda}(G) \leq (\tau/r_i) \]

**Upper Bound on Overhead Delay**

\[ \tilde{\gamma}(G) := [I_m(G^+) + \gamma(G^+)] - I_m(G) \]
**Min Rate: satisfiability**

Max delay (min rate) between two executions of $v_i$ occurs when the entire hierarchy is traversed with just one execution of the link operations that lead to $v_i$.

**Min Rate Constraints**

General case: involves two bounds

$$\bar{\lambda}(G) + \lambda(G) \leq \left( \frac{t}{r_i} \right)$$
**Min Rate Constraints (with ND operations)**

In the presence of ND operations in G:

- The latency $\lambda(G)$ needs to be bounded
- Relative rate constraints -- represented as a backward edge (i.e., max delay constraint) from G’s sink to source vertices => ND cycle in the constraint graph

**ND Operations: Data-dependent Loops**

- The loop index determines the number of times the loop body is invoked for each invocation of the loop link operation => delay of the loop operation is its loop index times the latency of the loop body
- If the constrained graph (G) contains at most one loop operation, $v$, on a path from source to sink
- The minimum rate constraint can be seen as a bound on the number of times the loop body (G) is invoked.

The minimum rate constraint can be seen as a bound on the number of times the loop body (G) is invoked.
**Satisfiability of Min Rate Constraints**

Consider a flow graph $G$ with an $ND$ operation $v$ representing a loop in the flow graph.

A minimum rate constraint $r_i$ on operation $v_i \in V(G)$ and $v_i \neq v$ is satisfiable if the loop index, $x_v$ indicating the number of times $G_v$ is invoked for each execution of $v$ is less than the bound $G(G) - 1 + \mu(v)$.

$$x_v := \left\lceil \frac{\tau \cdot r_i^{-1} - \gamma(G) - 1}{l_M(G_v)} + 1 \right\rceil$$

$\mu(v)$ represents the mobility of operation $v$, defined as the difference between the longest path that goes through $v$ and $l_M(G_v)$.

**Relative Min Rate Constraints**

Relative min rate constraint relative to $G$ is applied when $G$ is enabled and executing.

$$\overline{x_v} := \left\lceil \frac{\tau \cdot r_i^{-1} - \gamma(G) - 1}{l_M(G_v)} + 1 \right\rceil$$
1. Construct the Constraint Graph
   - add forward edges for *minimum delay* and *maximum rate* constraints
   - add backward edges for *maximum delay* and (relative) *minimum rate* constraints

2. Identify *cycles* by path enumeration for each of the *backward edges* in the constraint graph
   - check for constraint satisfiability, bound delays, etc.

3. Propagate *minimum rate* constraints up the graph hierarchy