Scheduling Kahn Process Networks

Paul Feautrier

Paul.Feautrier@ens-lyon.fr.

ENS Lyon

Plan

- What is a KPN?
- Why use KPNs? An Example.
- Scheduling KPNs, why and how?
- Scheduling constraints.
- Solving the Constraints.
- Provisional Conclusions.
- Technical Problems.
- Future Work.

What is a KPN?

- A set of independent, deterministic processes.
- A set of channels. Unbounded, FIFO, error-less.
- Non-blocking `send`, blocking `receive`.
- Each channel can have at most one producer and one consumer.

```
process three(int outport out) {
 int i;

 for(i=0;;i++){
 W1: send(out, 1);
 W2: send(out, 2);
 W3: send(out, 3);
     }
}
```

⇒

```
process four(int inport in) {
 int j;
 int k, t;

 for(j=0;; j++)
     for(k=0; k<3; k++)
 R:      t[k]=receive(in);
}
```

```
int channel c;

four(c);
three(c);
```

Why use KPNs?

- KPNs have deterministic channel behaviour.
  - The next best model after deterministic sequential programs.
- KPN have an intuitive graphical representation.
- Parallelism in KPNs: inter-process parallelism (control parallelism), intra-process parallelism (data parallelism).
- KPN can be executed, provided one has the required run-time system.
- Applications for KPN : signal processing, streaming, VLSI specifications.

Scheduling KPNs, Why and How?

- The programming model of KPNs is asynchronous.
- Most embedded systems have a synchronous execution model : SIMD, VLIW, systolic arrays.
- Synchronous systems are easier to specify, implement and verify.
- Synchronous systems can easily meet realtime constraints.
- Scheduling gives an opportunity to adjust the degree of parallelism.
- Absence of a schedule is an indication of deadlock.

What is a schedule?

- Notations : E the set of operations (execution of one statement) of a KPN. This set may be infinite, but if the programs in the processes have static control, a finite symbolic description can be constructed at compile time.
- A schedule is a function θ from E to N. $\theta(u)$ is the time at which operation u is initiated. The time is expressed in arbitrary units (e.g. clock cycles). The duration of operation u is noted $\partial(u)$.
- Since the set of operations is large (or even infinite), the schedule must be given in closed form.

What is the use of a schedule?

- By inverting a schedule, one obtains the set of operations to be initiated at any given time.
- This set must be finite, as there are only a finite number of resources (processors, functional units, ...) in the system.
- This set is a VLIW instruction, or a parcel for an EPIC machine, or should fit in a scheduling window on a superscalar processor.
- The object program can be constructed mechanically by *polyhedra scanning* techniques.
- What are the constraints a schedule must satisfy?

Dependence Constraints

- Processes in a KPN must have the same semantic before and after scheduling, hence, dependences must be satisfied.
- If the processes have static control, dependences can be represented as polyhedra.
- The dependence constraint (or causality condition) can be written:

$$\forall u, v \in E : u \; \delta \; v \Rightarrow \theta(u) + \partial(u) \leq \theta(v).$$

Message Constraints

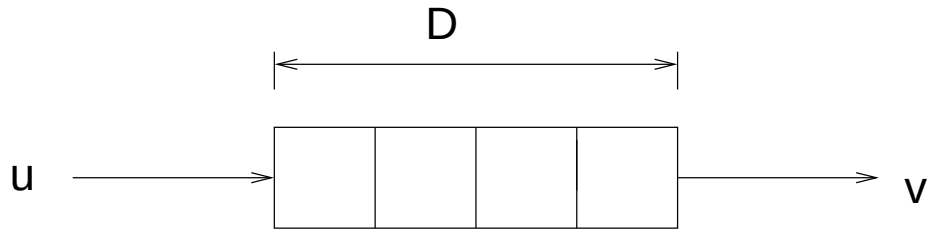
- Since the ultimate constraint is that the channel histories are not modified, the order of sends and receives must not be modified.
- A message cannot be received before it has been sent.
  Let $w(u)$ (resp $r(u)$) be the number of messages which have been sent (resp. received) *before* operation u. Let W_c (resp. R_c) be the set of sends (resp. receives) to channel c.

$$\forall u \in W_c, v \in R_c : w(u) = r(v) \Rightarrow \theta(u) + \partial(c) \leq \theta(v),$$

  where $\partial(c)$ is the transmission delay of channel c.

Capacity Constraints

- Infinite capacity channels are an abstraction which cannot be implemented, especially in embedded system.
- One may want to bound the number of unreceived message. This does not change the semantics, but it may generate deadlocks.



- Let D be the "depth" of channel c:

$$\forall u \in W_c, v \in R_c : r(v) \leq w(u) - D \Rightarrow \theta(v) \leq \theta(u).$$

- One may ignore the capacity constraints and find the necessary depth after scheduling, or try several values of D until a schedule is found.

Resource Constraints

- A resource is anything that can be lacking. In our case, resources are processors or functional units.
- In the abstract, a resource is a set of operations. $u \in \mathcal{R}$ means that operation u uses resource $\mathcal{R}$.
- In the simplest case, where all resources are distinct and all durations are 1, the resource constraint for $\mathcal{R}$ is:

$$\forall u, v \in \mathcal{R} : \theta(u) \neq \theta(v),$$

$$\forall u, v \in \mathcal{R} : u <_{\mathbf{seq}} v \Rightarrow \theta(u) + 1 \leq \theta(v) \vee \theta(v) + 1 \leq \theta(u).$$

Heuristics: ignore the second disjunct. The result has the same shape as a dependence constraint, but some solutions may be lost.

Solving the Constraints

- One cannot solve the scheduling problem unless one first select the shape of the schedules.
- In static control programs, an operation is named by giving the values of the surrounding loop counters. Schedule are chosen to be affine forms in these counters.
- All constraints have then the form:

$$\forall \boldsymbol{x} \in \boldsymbol{P} : \phi(\boldsymbol{x}) \geq 0.$$

  where $\boldsymbol{P}$ is a known polyhedron and ϕ is an unknown affine form. One first eliminate the quantifier by various techniques. The resulting problem is a linear programming problem and is solved by standard algorithms.
- One may take this opportunity to minimize interesting quantities, like the total duration of the program or latency.

Real Time Constraints

- KPNs are used to model systems with hard real time constraints. For instance, a video streamer must generate one screenful of pixels 30 times a second.
- This is easily done by fixing the value of some coefficients of the schedules (which are *periods*).
- The scheduler will either construct a schedule or decide none exists, e.g. if the operations delays are too long to sustain the required throughput.
- One may even express delays as a number of cycles times the clock period, and then maximize the clock period. The overall effect is to minimize power consumption.

Conclusions, Provisional

- Scheduling is a promising technique for compiling KPNs to VLIW or ASIC systems.
- The method was originally developped for high performance compilers and needs modifications for embedded systems.
- The central problem, which consists in finding the architecture with minimum “cost” for a given task, can be solved by a mixture of straightforward optimization and exploration.
- However, the method is still in its infancy. “Proof of concept” implementations are under development.

Technical Problems, I

Handling modulo and integer division

- Modulo and integer division occur quite frequently. For instance, a circular buffer of size S can be modeled as an array A with subscript `i%S`. These expressions are not linear, but can be linearized by a change of variable: $q = i \div S,\ 0 \leq i - q.S < S$.
- However, this is legitimate only if q is integral. In:

$$\forall x \in P : \phi(x) \geq 0,$$

  P has be replaced by its *integer hull*, by adding *cuts*.
- Experience shows that one well chosen cut is enough

Technical Problems, II

Resource Constraints

- If one associate a virtual variable to each resource, and if each operation that use this resource write into that virtual variable, then output dependences are created, with the end result of forbidding simultaneous use of this resource.
- If there are N resources of the same type, the virtual variable becomes an array of size N. This array must be subscripted by an expression $f(i) \bmod N$ where i is the iteration vector and f is linear.
- Experience shows that the quality of the result is very sensitive to the choice of f.
- The method is only a heuristics, and may lose some interesting solutions.

Technical Problems, III

Message Counting

- For the construction of message dependences, one needs to count messages, i.e. send or receive operations on a given channel.
- This can be done using P. Clauss methods. The results are Ehrhardt polynomials (i.e. polynomials with possibly periodic coefficients).
- Simpler methods may be used in many cases.
- Solving the message constraints is easy when counts are affine functions.
- Extension to the polynomial case is an open problem.

Technical Problems IV

Modular Scheduling

- Since the scheduling problem entails solving large linear programs, applying the method to real-life examples may be beyond the capacity of the best existing solvers.
- It would be nice if a specification could be split into several modules, if each module could be scheduled independently of other modules, with a final phase for fitting the module schedules.
- A candidate for a module might be several interconnected processes.
- It is by no mean obvious that in such a situation, the problem will stay within the limits of the polytope model.

Future Work

- Find a better way of handling resource constraints.
- Write code generators for many target systems (VLIW processor, systolic arrays, SystemC or VHDL).
- Bridge the gap between behavioral specifications and implementable specifications.
- Extend the method to hardware/software codesign and architecture exploration.