HW/SW Codesign

Exercise 3: Mapping and Partitioning (1/2)

11. October 2017

Andres Gomez
andres.gomez@tik.ee.ethz.ch
Reminder – System Synthesis

Mapping transforms behavior into structure and execution

- allocation: select components
- binding: assign functions to components
- scheduling: determine execution order
- ... finally, synthesis results into implementation

partitioning
mapping
Mapping = **Binding** + **Scheduling**

Bind software *(application)* components to hardware *(architecture)* components

How to specify?
How to specify - Application specification

- Depends on the model of computation
  Ex: task graphs, process network, state charts, ...

- Commonly represented as a graph - $G_P(V_P, E_P)$

- Nodes $V_P$ denote functional and communication units

- Edges $E_P$ denote data/control dependencies

Wireless LAN 802.11a job represented as a Synchronous Dataflow graph
How to specify — Architecture specification

• Depends on the model of platform

• Commonly represented as a graph - $G_A(V_A, E_A)$. Nodes $V_A$ denote architectural units. Edges $E_A$ denote links (connected-ness)

- Block diagram of a single tile (left) and the multi-tile SHAPES architecture (right). The availability of six links on the distributed network processor enables the construction of a scalable on-chip/off-chip toroidal network.
What then is binding

- Binding is a function $f : V_P \rightarrow V_A$.

- Some bindings may not be valid

  - Functionality constraints
    - Communication node cannot be bound to a DSP node
    - FFT decomposition node written for a DSP cannot be bound to a RISC processor node
    - Encryption node cannot be bound to a NoC switch

  - Connectedness constraints
    - Two application nodes requiring communication cannot be bound to two architectural nodes with no link between them
**Definition:** A *specification graph* is a graph $G_S=(V_S,E_S)$ consisting of a problem graph $G_P$, an architecture graph $G_A$, and edges $E_M$. In particular, $V_S=V_P \cup V_A$, $E_S=E_P \cup E_A \cup E_M$.
Problem 1

• Given
  – set of tasks,
  – set of processors
  – memory requirements $c_{i,j}^m$
  – energy requirements $e_{i,j}^m$

• To do
  – Formulate integer linear programs with different objectives and restrictions
Integer Linear Programming

An application is modeled as a set of tasks \( \{v_1 \ldots v_8\} \). We want to determine the binding of the tasks to two processors: \( p_1, p_2 \). Each task \( v_i \) comes with a memory requirement \( c_{i,1}^m \) and \( c_{i,2}^m \) on processor \( p_1 \) and \( p_2 \), respectively. Furthermore, each task \( v_i \) comes with an energy requirement \( c_{i,1}^e \) and \( c_{i,2}^e \) on processor \( p_1 \) and \( p_2 \), respectively.

3.1.a)
Specify an Integer Linear Program (ILP) for computing an energy-minimal partitioning.

3.1.b)
Extend the ILP from a) by enforcing that an equal number of tasks is assigned to each processor.

3.1.c)
Specify an ILP for computing a partitioning that respects both energy and memory requirements, but weighs the total memory usage twice as much as the total energy usage. The memory usage is the sum of all task memory requirements, and the total energy usage is the sum of the individual task energies.
3.1.d)
Extend the ILP from c) by enforcing that the total memory usage for processor $p_1$ does not exceed the (unitless) quantity $K$.

3.1.e)
Specify an ILP for computing a partitioning that minimizes the maximal memory usage for each of the two processors.
Problem 2

• Given
  – task graph,
  – architecture graph,
  – table of possible bindings

• To do
  – draw application graph with nodes for communication
  – specification graph
  – suggest modifications to architecture
Solution Problem 2
Solution Problem 2

- Restricted binding because of communication provided by architecture: no way to interconnect FPGA with MIPS

- Can try to improve the for instance interconnect bus1 and bus2 to a common shared bus or a hierarchical bus

Figure 1: Task graph and target architecture
Partitioning

- Partitioning problem is to divide a set of objects into mutually exclusive blocks (see formal definition in lecture slides)

- Several methods – ILP, random, hierarchical clustering, Kernighan-Lin algorithm, simulated annealing, Evolutionary algorithms

- Partitioning is a key step in binding decisions
  - What to run on software (RISC processor) and what to run on hardware (specialized co-processors)?
  - How to bind tasks on a multicore processor?
  - How to implement a given behavior on a FPGA?
Hierarchical clustering

• Define a **closeness function** between every pair of nodes
  – Designing closeness functions for real problem is quite an art. We will discover more on this in next exercise!

• Nodes that are close are good candidates for clustering into same partition

• Method:
  – in each step we cluster two **closest** nodes and **appropriately modify** the graph
  – After all steps, we **decide the cut-level** and generate the partition
Hierarchical clustering: lecture example

1. Merge two closest nodes
2. Modify the graph by changing the new weights using arithmetic mean
3. Repeat process till done
Hierarchical clustering: lecture example

- Choose cut line and generate partitioning
- Another art for real problems

step 1:

step 2:

step 3:
Problem 3

• Given
  – A graph with closeness functions for each pair of connected nodes

• To
  – Hierarchically cluster the graph by setting closeness functions of new edges using
    – average values
    – minimum values

Figure 2: Graph with objects
Solution Problem 3

Figure 4: Hierarchical Clustering with "average closeness"
Solution Problem 3

Figure 5: Hierarchical Clustering with "minimum closeness"