ECE 695R: System-on-Chip Design

Module 2: HW/SW Partitioning
Lecture 2.10: HW/SW Co-Synthesis: An ILP Formulation I

Anand Raghunathan
raghunathan@purdue.edu

Fall 2014, ME 1052, T Th 12:00PM-1:15PM
HW/SW Co-synthesis: An ILP Formulation

- An Integer Linear Programming (ILP) formulation of HW/SW partitioning and scheduling
  - Why together?
    - Highly interdependent

- **Note:** ILP formulations are usually not very scalable, but offer a platform to mathematically formulate a problem and understand its mathematical structure

Background – Integer Linear Programming

- A mathematical framework for expressing optimization problems
  - **Variables** or unknowns are integers
  - Minimize **cost** function subject to **constraints**
    - Cost function is a **linear** function of variables
    - Constraints are **linear** inequalities in variables

Cost function: \[ C = \sum_{x_i \in X} a_i x_i \text{ with } a_i \in \mathbb{R}, x_i \in \mathbb{N} \]

Constraints: \[ \forall j \in J : \sum_{x_i \in X} b_{i,j} x_i \geq c_j \text{ with } b_{i,j}, c_j \in \mathbb{R} \]
Background - Integer Linear Programming

• If the variables are constrained to be \{0, 1\}, the problem is called 0/1 ILP or Binary Integer Programming (BIP)

• ILP is NP-Hard
  – 0/1 ILP also NP-Hard, but more advanced algorithms exist
  – State-of-the-art ILP solvers can handle instances of up to a few thousand variables (greatly depends on problem structure)

Formulating a problem as ILP is often a good starting point even if heuristics are eventually used to solve it
HW/SW Partitioning as ILP: Basics

• Assumptions
  – Program represented as “task-level” control/data flow graph (task graph)
  – Nodes represent computations (functions), edges represent data communicated between the functions
  – Execution of task graph can be statically scheduled in time
HW/SW Partitioning as ILP: Basics

• Assumptions
  – Target Architecture consists of a fixed number of symmetric programmable processors and arbitrary no. of HW accelerators
    • Multiple processors generalizes partitioning into Partitioning + Mapping
  – Each accelerator can contain an arbitrary amount of hardware
HW/SW Partitioning as ILP: Inputs

• Nodes in the task graph
  – $N_v$ nodes: $v_1 \ldots v_{N_v}$
  – $e_{ij} = 1$ iff there is an edge $v_i \rightarrow v_j$

• Costs and execution times for tasks
  – $C^{\text{HW}}_i = \text{Hardware cost for task } i$ (if implemented as an accelerator)
  – $C^{\text{SW}}_i = \text{Program + data memory cost for task } i$ (if implemented as SW)
  – $T^{\text{HW}}_i = \text{execution time of task } i$ in HW
  – $T^{\text{SW}}_i = \text{execution time of task } i$ in SW
  – Assume communication times are lumped into execution times (can easily extend to model separately)