Embedded System Design and Modeling

EE382N.23, Fall 2017

Homework #4

Mapping and Exploration

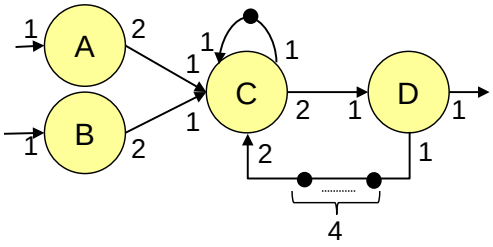
Assigned: November 6, 2017
Due: November 20, 2017

Instructions:

- Please submit your solutions via Canvas. Submissions should include a single PDF with the writeup and a single Zip or Tar archive for any supplementary files (e.g. source files, which has to be compilable by simply running 'make' and should include a README with instructions for running each model).
- You may discuss the problems with your classmates but make sure to submit your own independent and individual solutions.
- Some questions might not have a clearly correct or wrong answer. In general, grading is based on your arguments and reasoning for arriving at a solution.

Problem 4.1: SDF Mapping

Given the SDF graph from Homework #2, Problem 2.2(c), which is repeated below (with n fixed to its minimal value and m=2n.), explore possible executions of the graph on a homogeneous multi-processor architecture template with up to 4 processors. Assume that all processors are identical, all actors require one time unit per firing to execute, and all buffers are stored in a shared memory with zero communication overhead. Find mappings and periodic schedules of the graph to minimize cost (= number of processors and memory size needed), maximize throughput (= sustained rate of iterations), and minimize latency (= time required for one complete iteration of the graph).



(a) In class, we discussed that every SDF graph can be converted into an equivalent homogeneous SDF (HSDF) model according to its repetition vector. This is called the SDF model's precedence graph. Show the precedence graph for this example.

(b) Write down a single-processor schedule that minimizes buffer cost for this example. What is the minimally required memory size for execution of the graph? What is the achievable throughput and latency using a single processor?

(c) Apply a list scheduling algorithm to map and schedule one iteration of the precedence graph onto 2 processors. Use the level of a node in the graph (longest distance, i.e. maximum number of nodes on the path to the sink, i.e. to any node with no successors in the same iteration) as priority function for the list scheduler. Show the step-by-step operation of the algorithm (state of the ready list, priorities and mapping decisions made in each step), as well as the final schedule. You can assume that different firings of the same actor can be mapped

on different processors. Note: as discussed in class, an iteration is complete once the graph is back to its initial token state. What is the latency and throughput of your schedule?

(d) Assume that you can schedule the graph in a pipelined fashion, where one or more consecutive iterations overlap, i.e. a new iteration can begin while previous iterations are still running. Show the schedules with minimal latency and maximal throughput on 2 processors. You can assume that different firings of the same actor can be mapped on different processors. What is the latency and throughput in each case?

(e) Can a smaller latency and/or higher throughput be achieved on more than 2 processors? If so, how many processors are needed to achieve smallest latency? What is the highest throughput and corresponding minimal latency for a pipelined scheduling on up to 4 processors? Show the corresponding schedules including both latency and throughput in each case.

(f) Write down the ILP formulation for the mapping (combined partitioning and scheduling) of the problem graph on 2 processors. Limit the optimization problem to a basic (non-pipelined) schedule of a single iteration of the graph in the time window $0 \leq t < T_{max}$. List all the inputs to your ILP and all constraints for number of iterations, unique mapping of actors to processors, sequential execution on each processor and sequencing/dependency relations between actors. Formulate an objective to minimize overall latency (time to execute the single iteration of the graph).

Encode your ILP formulation in the LP file format and feed it into either the `lp_solve` or `glpsol` solver (both are installed on the LRC machines). Show the solution found by the solver and verify that it is a valid mapping and schedule. Make sure to submit your `.lp` file and the output file generated by the solver for this assignment.