Embedded System Design and Modeling

EE382V, Spring 2014

Homework #2

Models of Computation (MoCs)

Assigned: February 18, 2014
Due: February 27, 2014

Instructions:

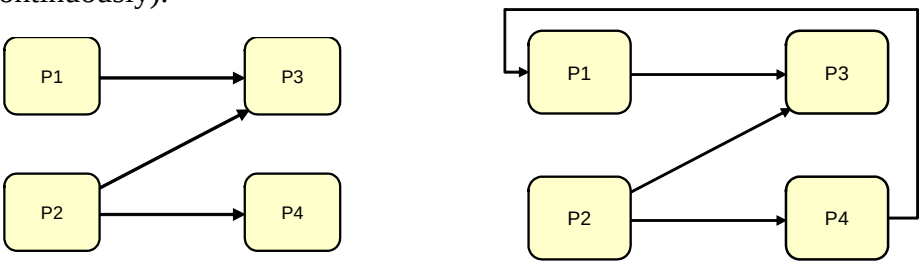
- Please submit your solutions via Blackboard. Submissions should include a single PDF with the writeup and a single Zip or Tar archive for any supplementary files (e.g. source files, which has to be compilable by simply running 'make' and should include a README with instructions for running each model).
- 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 2.1: Models of Computation and Languages

In class, we learned about different Models of Computation (MoCs) for system specification: KPN, SDF, FSM(D), HCFSM. We also discussed that such higher-level models can be mapped down to a basic discrete-event (DE) MoC for simulation. Briefly sketch (show code snippets) how each of the above MoCs can be represented on top of SpecC. Discuss in how far the SpecC model is semantically equivalent to the original MoC. Apart from DE-driven simulation/execution, how well do languages such as SpecC (or SystemC) support capturing such models? Specifically, think of analysis and synthesis (e.g. scheduling) of captured models by design automation tools.

Problem 2.2: Kahn Process Networks (KPN)

(a) Consider the two variants of the KPN example we discussed in class. Again, assume P3 does not consume any tokens on the P2→P3 arc, while all other processes service their ports in a round-robin fashion (such that there is no deadlock and the KPN is supposed to run continuously):



For each of the two examples, is there a scheduling strategy that is non-terminating (free of artificial deadlocks), complete, bounded or any combination thereof? If so, what is the corresponding strategy each? What can you conclude about the tradeoffs between non-termination, completeness and boundedness in particular instances of models? Are there combinations of properties that are related or are there always choices to be made, and if so, under what conditions (cases of KPN models)?

(b) In class, we mentioned that Parks' algorithm is not guaranteed to find a complete, bounded and non-terminating schedule even if one exists. Show an example of a KPN

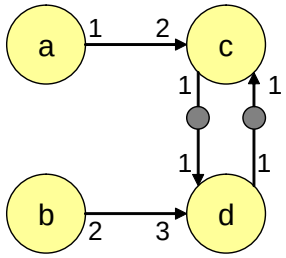
where such a schedule exists but Parks' algorithm fails to find it. Hint: in the KPN example presented in class, think about token patterns that can happen on the P2→P3 edge.

(c) Can every dataflow model be executed as a Kahn Process Network (KPN)? Why or why not? If yes, what would be the advantages and disadvantages of executing a SDF graph as a KPN? If not, under what conditions can it not?

(d) Can every KPN be converted into an equivalent dataflow model? Why or why not? If yes, what would be the advantages and disadvantages? If not, under what conditions is a conversion not possible?

Problem 2.3: Dataflow

For the SDF graph on the right:



(a) Show that the graph is consistent, and that it has a valid schedule. What is the repetition vector of the graph?

(b) It can be shown that every SDF graph can be converted into an equivalent homogeneous SDF (HSDF) model. In the so-called precedence graph, every actor firing in the repetition vector is turned into a dedicated homogeneous actor instance (e.g. firings *A1*, *A2*, … of an SDF actor *A*). Show the precedence graph for this example. What can you say about the complexity (number of nodes) of the precedence graph as a function of the size of the original SDF graph?

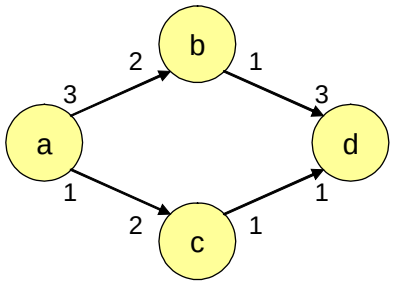
(c) Are both of the initial tokens necessary? What is the minimum number of initial tokens needed for this graph to be consistent and to have a valid schedule? Does adding or removing tokens change the consistency or validity of the graph? Assuming actor *d* produces the external output of the graph, does the behavior and precedence graph change, and if so, how?

(d) List all possible minimal periodic static schedules for one iteration of the graph.

(e) Find the periodic schedule with the lowest token buffer usage. What is the minimum buffer usage?

(f) Assume each actor firing executes in one time unit. Find the schedule with the highest throughput (output token rate, i.e. average number of firings of the output actor *d* per time unit). What is the maximum throughput on a single processor? What is the maximum throughput you can achieve on two processors?

For the SDF graph on the right:

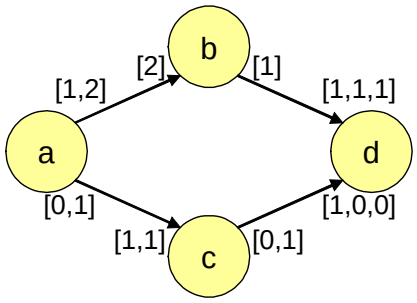


(g) Show that the graph is consistent and that it has a valid schedule. What is the repetition vector of the graph?

(h) Show the precedence graph for this SDF.

(i) Find the periodic single-processor schedule with the lowest buffer usage. What is the minimally required buffer space?

For the CSDF graph on the right:



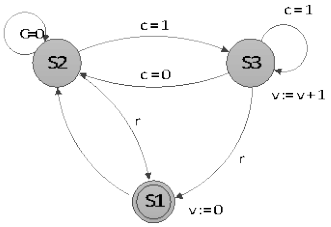
(j) Assuming that within each periodic iteration of the graph, the number of firings of each actor must be an integer multiple of the total number of phases/firings in the actor's cycle, show that the graph is consistent and that it has a valid repetitive schedule. What is the repetition vector of the graph?

(k) Show the precedence graph for this CSDF.

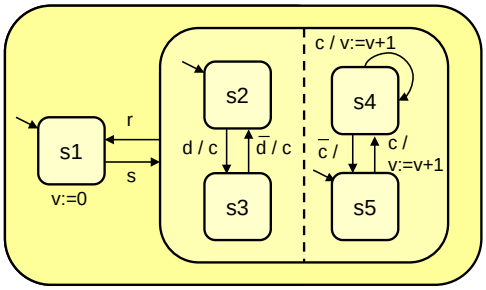
(l) Find the periodic single-processor schedule with the lowest buffer usage. What is the minimally required buffer space? What are the differences in schedulability for this graph compared to the similar SDF graph above?

Problem 2.4: State Machines

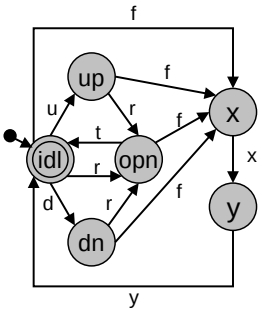
In class, we have discussed the concepts of extended state machines (FSMs with data), hierarchy (OR state) and concurrency (AND state) for managing complexities in FSMD and HCFSM models.



(a) Convert the extended FSM (EFSM)/FSMD on the right into an equivalent FSM. Can every FSMD be represented as a FSM? What can you say about the complexity (number of states) of the FSM as a function of the size and parameters of the original FSMD?



(b) Convert the HCFSM(D) on the right into an equivalent FSMD. Note that the concurrent (AND) composition of states is communicating through signal *c*, and that the HCFSM has Mealy semantics. Transitions for unspecified input conditions default to remaining in the same state, where unspecified outputs default to absence of producing any event. Absence is otherwise indicated as negated conditions on signals. What functionality does the state machine actually perform?



(c) Try to reduce the complexity of the FSM on the right by converting it into a HCFSM using hierarchy (OR states) and/or concurrency (AND states) wherever possible. What can you generally say about the complexity (number of states and transitions) of an FSM as a function of the size of an equivalent HCFSM?