Embedded System Design and Modeling

EE382N.23

Previous Exam and Homework Problems

Part 1: Languages

Problem 1.1: Languages (10 points)

(a) Why are sequential programming languages (e.g. C/C++/Java) considered to be insufficient for embedded system specification and design?

(b) Why are hardware design languages (e.g. VHDL/Verilog) considered to be insufficient for embedded system specification and design?

Problem 1.2: Models of Computation and Languages (15 points)

In class, we learned about the different Models of Computation (MoCs): KPN, SDF, FSM(D), HCFSM, PSM.

(a) What is the relationship between MoCs and languages?

(b) Can SpecC support all these MoCs? If so, briefly sketch for each MoC that you think can be supported how you would represent a corresponding model in SpecC.

(c) Specifically show a code template for how you would write an SDF actor and an SDF model in SpecC. Discuss in how far the SpecC code is equivalent to the original SDF model, e.g. in terms of being amendable to automated analysis or synthesis by design tools.

Problem 1.3: Language Concepts (20 points)

We have discussed the concepts of synthesizability, orthogonality and separation of concerns in languages.

(a) What are the requirements for languages to be synthesizable?

(b) Briefly define what orthogonality is? What is separation of concerns?

(c) Try to show an example other than the ones discussed in class of non-orthogonal concepts or constructs in a language.

(d) Name two concerns that should be separated as they cover orthogonal aspects? Show a short code excerpt that demonstrates a non separated code and code that separates these concerns (in C/C++/SpecC/SystemC or similar).

Problem 1.4: Discrete-Event Semantics (35 points)

For each of the following code examples, what is the value of `myB` printed at the end of execution and at what simulated time does the program terminate. You are free to run the code on top of the SpecC simulator and observe the program output, but you need to provide an explanation and reasoning of why the program is behaving as it is (e.g. sequence of events happening during simulation):

(a)

```
behavior A(int myB)
{
  void main(void)
  {
    myB = 10;
  }
};

behavior B(int myB)
{
  void main(void)
  {
    myB = 42;
  }
};

behavior Main(void)
{
  int myB;

  A a(myB);
  B b(myB);

  int main(void) {
    par { a; b; }
    printf("%d", myB);
    return 0;
  }
};
```

(b)

```
behavior A(int myB)
{
  void main(void) {
    waitfor 42;
    myB = 10;
  }
};

behavior B(int myB)
{
  void main(void)
  {
    myB = 42;
  }
};

behavior Main(void)
{
  int myB;

  A a(myB);
  B b(myB);

  int main(void) {
    par { a; b; }
    printf("%d", myB);
    return 0;
  }
};
```

(c)

```
behavior A(int myB)
{
  void main(void) {
    myB = 10;
    waitfor 42;
  }
};

behavior B(int myB)
{
  void main(void) {
    waitfor 10;
    myB = 42;
  }
};

behavior Main(void)
{
  int myB;

  A a(myB);
  B b(myB);

  int main(void) {
    par { a; b; }
    printf("%d", myB);
    return 0;
  }
};
```

(d)

```
 1 behavior A(int myA, event e)
 2 {
 3   void main(void) {
 4     myA = 10;
 5     notify e;
 6     myA = 11;
 7     notify e;
 8     waitfor 10;
 9   }
10 };
11
12 behavior B(int myA, int myB, event e)
13 {
14   void main(void) {
15     wait e;
16     myB = myA;
17   }
18 };
19
20 behavior Main(void)
21 {
22   int myA;
23   int myB;
24   event e;
25
26   A a(myA, e);
27   B b(myA, myB, e);
28
29   int main(void) {
30     par { a; b; }
31     printf("%d", myB);
32     return 0;
33   }
34 };
```

(e)

```
behavior A(int myA, event e)
{
  void main(void) {
    myA = 10;
    notify e;
    waitfor 10;
    myA = 11;
    notify e;
  }
};

behavior B(int myA, int myB, event e)
{
  void main(void) {
    wait e;
    myB = myA;
  }
};

behavior Main(void)
{
  int myA;
  int myB;
  event e;

  A a(myA, e);
  B b(myA, myB, e);

  int main(void){
    par { a; b; }
    printf("%d", myB);
    return 0;
  }
};
```

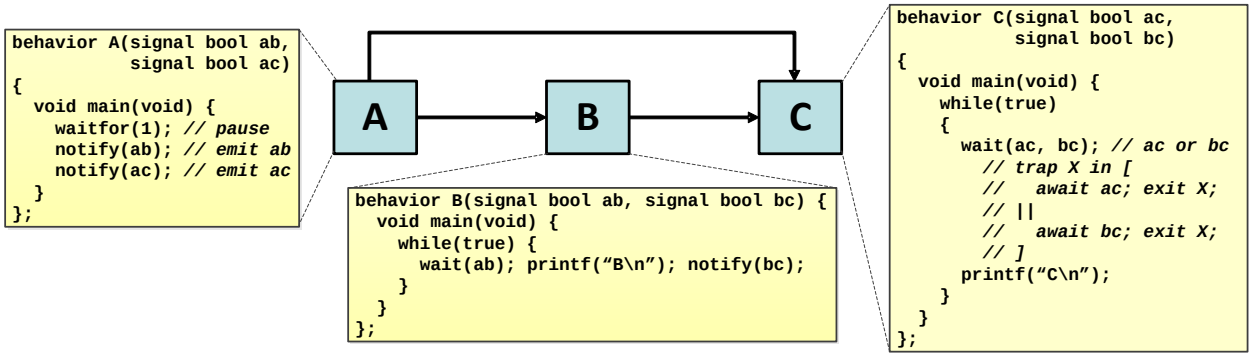
(f) What code has to be inserted at the beginning of behavior `B` (line 15) in (e) to change the output of the program? Give two different options. What must *not* appear there for the program not to deadlock?

(g) Why do SpecC events have a semantic in which they can get lost? Under what condition do SpecC events get lost? What type of channel/communication could not be modeled if delivery would always be guaranteed?

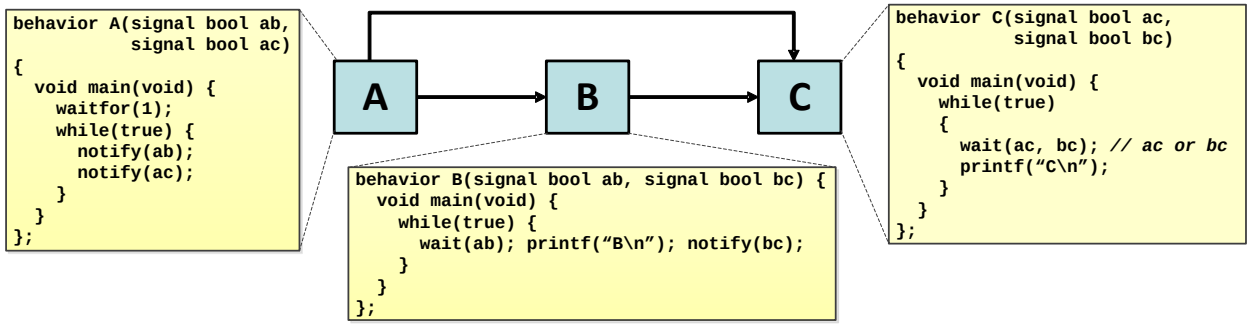
Problem 1.5: Language Semantics (25 points)

For each of the following examples, what is the output of the program assuming (i) discrete-event (with delta cycle) and (ii) synchronous-reactive (with fixed-point) semantics for the signal interactions between concurrent blocks? For (i), you are free to run the examples in the SpecC simulator, but you need to provide an explanation of the behavior. For (ii), examples of equivalent Esterel syntax are indicated in the comments in some cases (and same conditions apply: you can feed equivalent code though an Esterel compiler and simulator, but you need to provide explanations).

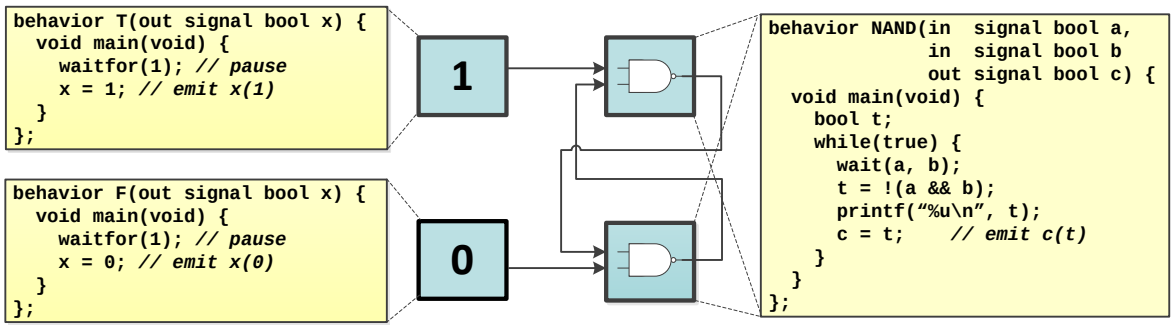
(a)



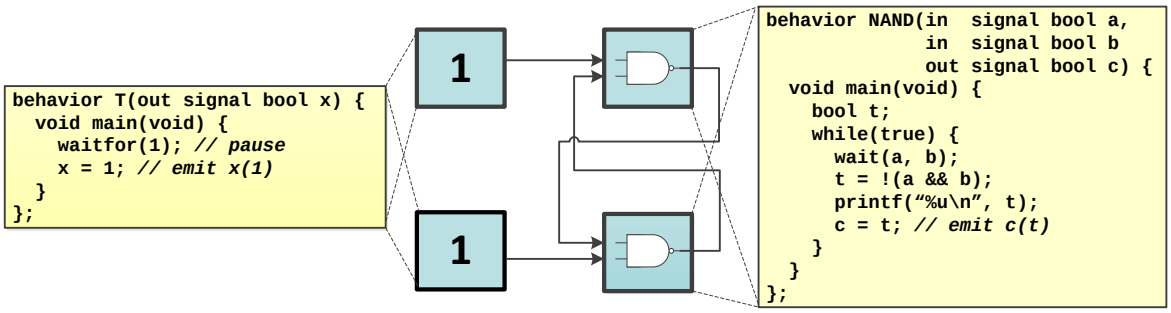
(b)



(c) You can assume that (valued) signals have a default value of '0':



(d) You can assume that (valued) signals have a default value of '0':



(e) Ignoring other sources of non-determinism, e.g. coming from the sequential code inside the blocks themselves (through the C language), is a discrete-event model that does not allow shared variables and only supports (valued) signals for communication deterministic? If so, why? If not, what sources of non-determinism still exist? What can you say about the (non-)determinism of (valued) signals in a synchronous-reactive language?

Embedded System Design and Modeling

EE382N.23

Previous Exam and Homework Problems

Part 2: Models of Computation

Problem 2.1: Non-determinism (10 points)

(a) What is nondeterminism?

(b) How might nondeterminism arise? (give one example not discussed in class)

(c) What are the advantages and disadvantages of having nondeterminism in a language or model, i.e. in what circumstances might it be positive/desired or negative/undesired?

Problem 2.2: Process and Dataflow MoCs (15 points)

In relation to process network and dataflow type Models of Computation (MoCs), we talked about properties such as determinism and deadlocks. For the following questions, either provide a proof (property holds in all cases) or a counter-example:

(a) Does adding an `empty()` method that reports whether a channel has waiting tokens change the determinism of Kahn Process Networks (KPNs)?

(b) To determine the relative rates for a static schedule of Synchronous Dataflow (SDF) actors, one can derive and solve the system of linear *balance equations*. Can a SDF model have satisfiable balance equations (i.e. be consistent) and not run forever? Can a SDF model violate the balance equations (i.e. be inconsistent) and still run forever?

(c) Can a tree-structured (i.e. acyclic) SDF model ever have deadlocks? Can a static, complete and bounded schedule always be found?

Problem 2.3: Kahn Process Networks (KPN) (15 points)

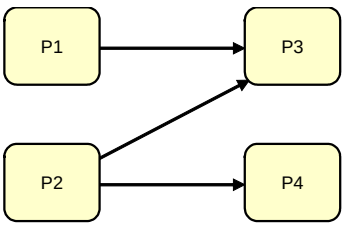
(a) Does Parks' KPN scheduling algorithm always find a bounded schedule if it exists? Why or why not?

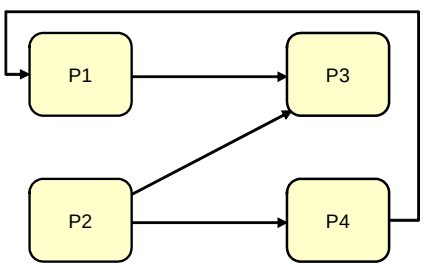
(b) Does Parks' algorithm always find a complete KPN schedule, if it exists? Why or why not?

(c) Does Parks' algorithm always find a non-terminating (free of artificial deadlocks) schedule, if it exists? Why or why not?

Problem 2.4: Kahn Process Networks (KPN) (25 points)

(a) Consider two variants of the KPN example 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)?

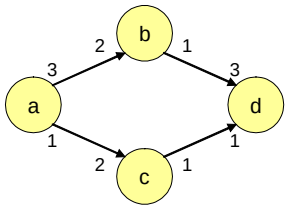
(b) In class, we mentioned that Parks' algorithm is not guaranteed to find a complete, bounded and non-terminating schedule even if one exists. Show an example of a KPN where such a schedule exists but Parks' algorithm fails to find it. Hint: in the KPN example presented in class, think about token patterns 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 2.5: Dataflow Synthesis (25 points)

For the SDF graph on the right:



(a) Show that the graph consistent and that it has a valid schedule.

(b) In class, we discussed that an SDF graph can always be converted into an equivalent homogeneous SDF (HSDF) model based on its repetition vector, where the resulting HSDF model is called the precedence graph of the original SDF model. In the 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) List all possible minimal periodic static schedules.

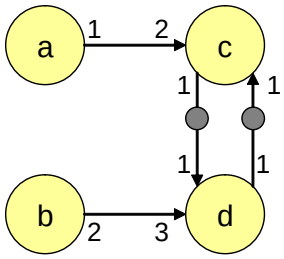
(d) Find the periodic schedule with the lowest token buffer usage. What is the minimum buffer usage?

(e) 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?

(f) Assume the graph is scheduled on two processors where each actor executes in one time unit on either PE and buffers are stored in a shared memory with zero communication overhead. Find a fixed assignment of actors to PEs and a corresponding schedule that maximizes throughput. What is the maximum throughput on two processors?

Problem 2.6: Synchronous Dataflow (15 points)

For the SDF graph on the right:

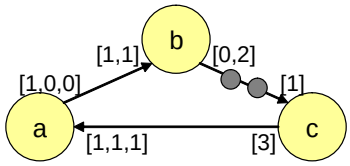


(a) Show that the graph is consistent, and that it has a valid schedule. What is the repetition vector of the graph? Give a minimal periodic static schedule if one exists.

(b) 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?

Problem 2.7: Cyclo-Static Dataflow (15 points)

For the CSDF graph on the right:



(a) Assuming that within each periodic iteration of the graph, the number of firings of each actor must be an integer multiple of the 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?

(b) Show the precedence graph for this example.

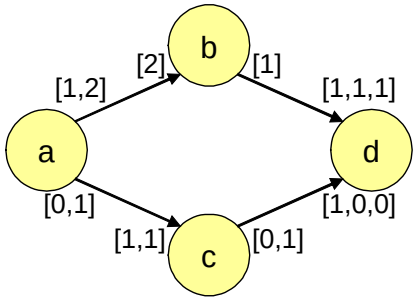
(c) List all possible minimal periodic static schedules.

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

(e) Find single-processor and dual-processor schedules with maximal throughput. What are the buffer requirements in each case?

Problem 2.8: Cyclo-Static Dataflow (15 points)

For the CSDF graph on the right:



(a) 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?

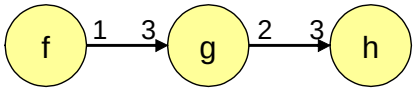
(b) Show the precedence graph for this CSDF.

(c) 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?

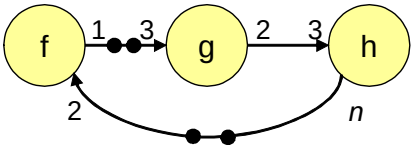
Problem 2.9: Synchronous Dataflow (SDF) (15 points)

For each of the following SDF models, determine whether a valid schedule exists that can be executed repeatedly and indefinitely in bounded buffer memory. Write down the balance equations, determine if and when the graph is consistent and give the repetition vector for a minimal (least integer) schedule. If it exists, write down a schedule that minimizes buffer sizes. How much buffer space is needed on each communication channel?

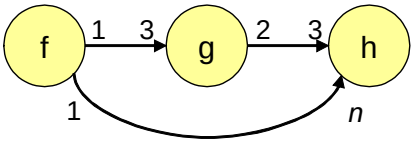
(a)



(b) *n* is an integer parameter.

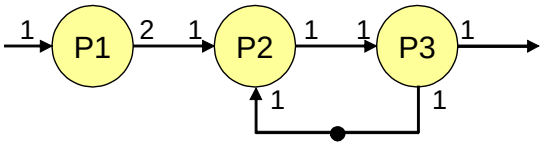


(c) *n* is an integer parameter.



Problem 2.10: Process and Dataflow Composability (10 points)

(a) In the following synchronous dataflow graph, which pairs of connected actors, if any, can be composed into a hierarchical super-actor to give a valid SDF graph that consists of the super-actor and the remaining actor (show the corresponding hierarchical graph)? For any invalid pairs, show a valid hierarchical graph in which the composite super-actor is converted into an appropriate CSDF actor instead.



(b) Are KPN models composable? In the example above, when executed as KPN, can every pair of connected processes be composed into a hierarchical super-process while maintaining KPN semantics?

Problem 2.11: State-Machine MoCs (15 points)

In class, we have discussed the concepts of hierarchy (OR state) and concurrency (AND state) for reducing complexities in a HCFSM (e.g. StateCharts) model. However, both hierarchical and concurrent FSM compositions can be converted into an equivalent plain FSM model:

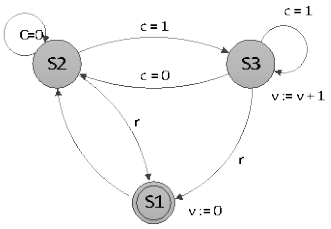
(a) Show an example of converting a hierarchical FSM into an equivalent plain FSM. Derive an expression for the complexity (number of states and number of transitions) of the equivalent plain FSM as a function of the complexity of the OR-composed FSMs.

(b) Show an example of converting a concurrent FSM into an equivalent plain FSM. Derive an expression for the complexity (number of states and number of transitions) of the equivalent plain FSM as a function of the complexity of the AND-composed FSMs.

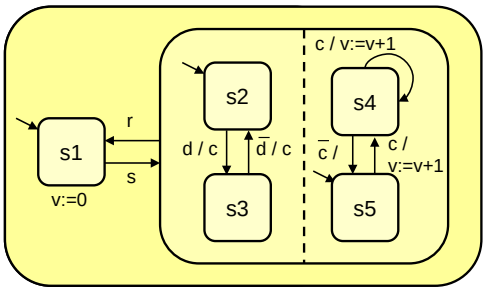
Problem 2.12: State Machines (15 points)

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.

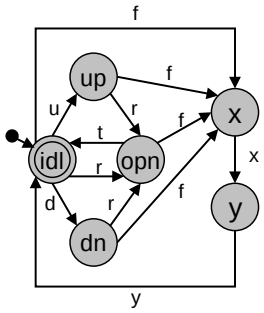
(a) Convert the extended FSM (EFSM)/FSMD on the right into an equivalent FSM. Can every FSMD be represented as a FSM? What can you say about the complexity (number of states) of the FSM as a function of the size and parameters of the original FSMD?



(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?



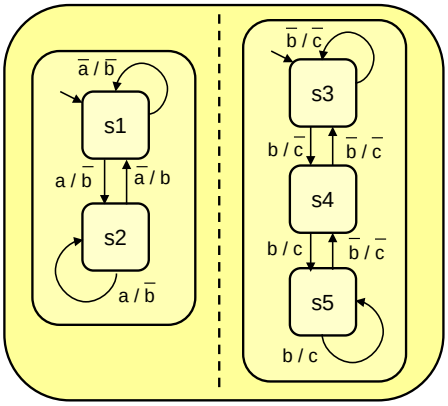
(c) Try to reduce the complexity of the FSM on the right by converting it into a HCFSM using hierarchy (OR states) and/or concurrency (AND states) wherever possible.



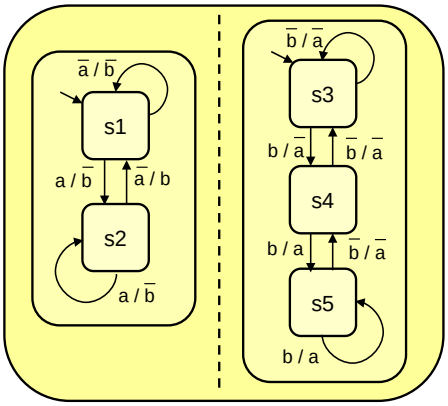
Problem 2.13: State Machine Models (20 points)

Convert the following Mealy HCFSM compositions into an equivalent single, flat FSM, if possible. Indicate which states of the composition are unreachable, if any. Note that presence/absence of signals as indicated by the x / $\overline{x}$ notation is really the same as setting their value to '1' or '0', respectively.

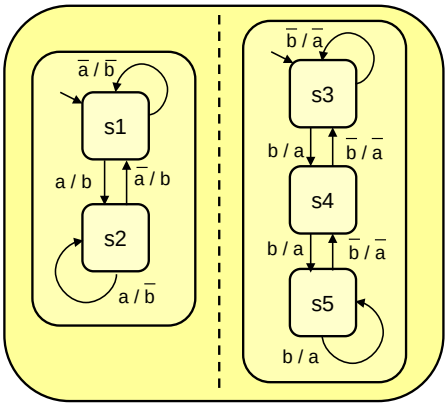
(a)



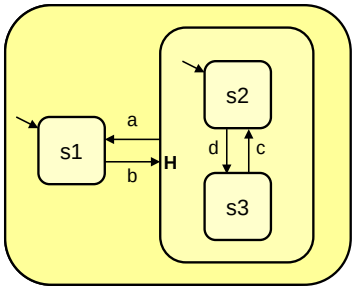
(b)



(c)



(d) The Statecharts realization of HCFMs has an additional construct called *history transitions* (marked by an 'H'), which, when taken, resume a hierarchical destination in whatever state it was last in (or its initial state on the first entry). Convert the following history-based hierarchical FSM into an equivalent regular flat FSM:



Embedded System Design and Modeling

EE382N.23

Previous Exam and Homework Problems

Synthesis, Refinement, Mapping & Exploration

Problem 3.1: Design Methodology (10 points)

(a) Compare and contrast a top-down, bottom-up and meet-in-the-middle (platform based) design methodology.

(b) Why do we perform computation design before communication design in the SpecC/SCE methodology?

Problem 3.2: Modeling and Refinement (10 points)

(c) List all the design decisions that influence and are represented in the computation (scheduled architecture) model. What effects of these decisions can be observed and validated (e.g. through simulation) at that level?

(d) List all the design decisions that influence and are represented in the communication model (PAM or TLM). What effects of these decisions can be observed and validated (e.g. through simulation) at that level? What is the advantage and disadvantage of a TLM over a PAM in that respect?

Problem 3.3: System Design (10 points)

During design space exploration as part of the system design process, the target system architecture and its key architectural parameters are decided on. These design decisions have a major influence on the final design quality metrics such as (i) performance, (ii) power, (iii) cost, and (iv) time-to-market:

(a) Briefly discuss how the following target platform styles rate in relation to each other in terms of the metrics listed above:
- A pure software solution on a general-purpose processor
- A general-purpose processor assisted by a custom hardware accelerator/co-processor
- A general-purpose processor and a specialized processor (DSP or ASIP or GPU)

(b) Try to sketch a potential simple strategy for exploring the design space for a given application under a given set of constraints/requirements.

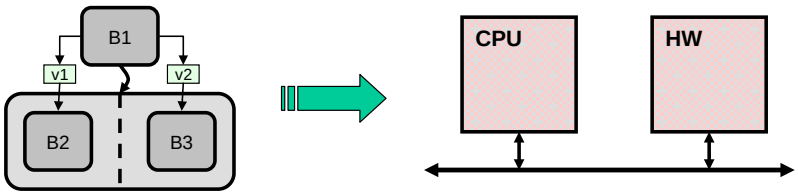
Problem 3.4: Amdahl's Law (10 points)

(a) Derive the equation for the theoretical upper bound on the speedup S that can be obtained in a sequential program when only a fraction P of the program can be accelerated (e.g. through implementation in custom hardware) by a maximum speedup factor of A (a speedup of $A = 2$ means fraction P can be run twice as fast).

(b) What is S for $P = 60\%$ and $A = 10$? What is S for $P = 60\%$ and $A = \infty$ (running the accelerated portion in zero time)? What is your interpretation of the results?

Problem 3.5: Design Space Exploration (20 points)

Consider a simple system consisting of three behaviors *B1*, *B2* and *B3* communicating through variables *v1* and *v2*, as shown on the left. It is supposed to be implemented on an architecture consisting of two PEs, *CPU* and *HW*, connected via a single bus, as shown on the right:



Assume that execution times of behaviors on PEs and sizes of variables are as shown in the following table (1 kB = 1000 byte). Behaviors assigned to the same PE must be scheduled sequentially on either PE.

	B1	**B2**	**B3**	**v1**	**v2**
CPU	3 ms	12 ms	20 ms	10 kB	50 kB
HW	2 ms	6 ms	15 ms		

Furthermore, assume that communication through the local memory inside a PE happens in zero time. For communication of variables that cross PEs, we implement a distributed, message-passing variable mapping scheme where the bus has a bandwidth of 10 Mbyte/s.

(a) What is the total delay if all behaviors are mapped to the *CPU*?

(b) What is the total delay if *B1* and *B2* are mapped to the *CPU* and *B3* is mapped into *HW*?

(c) What is the total delay if *B1* and *B3* are mapped to the *CPU* and *B2* is mapped into *HW*?

(d) Is there a mapping that runs faster than either of (a), (b) or (c)? If not, why not? If yes, what is the mapping and delay?

Problem 3.6: Partitioning and Scheduling (20+10 points)

In class, we introduced an ILP formulation for multi-processor partitioning and scheduling of synchronous dataflow (SDF) graphs. We also talked about the fact that every SDF graph can be converted into an equivalent homogeneous SDF (HSDF) model, and we introduced ILP formulations for general task partitioning and scheduling. Using an SDF to HSDF transformation as a preprocessing step, we can develop a simplified ILP formulation for partitioning and scheduling of converted HSDF graphs:

(a) Develop an ILP model for multi-processor partitioning and scheduling of HSDF graphs. Limit the schedule to execution of a single iteration of the graph in the time window $0 \leq t \leq T$, i.e. do not take overlapping/pipelining across iterations into account. Assume each processor has the same cost and each actor takes 1 time unit to execute independent of where it is mapped to. Your ILP formulation should only have two types of binary (i.e. $\in \{0,1\}$) decision variables:

A_{ij}: Actor i mapped to processor j

s_{it}: Actor i starts execution in time t

List all the inputs to your ILP and develop constraints for number of iterations, unique mapping of actors to processors, sequential execution on each processor, and sequencing/dependency relations between actors. Finally, formulate an objective to minimize overall cost and latency (time to execute the single iteration of the graph).

(b) Apply the ILP to the SDF model from Part 2, Problem 2.5 when scheduled on two processors (as in 2.5(f)). Write down the specific constraints and objective functions for this problem instance. Show that your answer for 2.5(f) is a valid solution to the ILP.

(c) Extra credit: without introducing additional decision variables, how could execution times ≥ 1 time unit that vary from actor to actor (but not for different processors) be taken into account?

Problem 3.7: Mapping and Exploration (20 points)

Given the SDF graph from Part 2, Problem 2.6, explore various approaches for an automated and optimized mapping of this graph onto a 2-processor system. Assume that processor are homogeneous and that each actor takes 1 time unit to execute independent of where it is mapped to. Remember that every SDF graph can be converted into an equivalent homogeneous SDF model. Using this SDF to HSDF transformation as a preprocessing step, we can apply mapping algorithms developed for standard task graphs to the equivalent precedence graph of the SDF:

(a) 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 $0 \leq t \leq T$. 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).

Finally, write down one valid (not necessarily optimal) solution of the ILP and its latency.

(b) Apply a list scheduling algorithm that uses the level (longest distance to the sink, i.e. critical path length) of a node in the graph as priority function. Show the step-by-step operation of the algorithm (state of the priority-sorted ready queue and mapping decisions made in each step), as well as the final graph execution generated by the scheduler as part of your writeup.

As discussed in class, under the assumptions of this assignment (uniform tasks and processors), a highest-level first (HLF) list scheduler as applied here is the same as Hu's algorithm, which is provably optimal for such problems. Compare your list scheduling result to the ILP result in (a), and confirm that it is a valid and optimal mapping (i.e. either better or the same than the mapping solution you obtained in (a)).