Embedded System Design and Modeling

ECE382N.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.

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

(b) Can SystemC 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 SystemC.

(c) Specifically show a code template for how you would write an SDF actor and an SDF model in SystemC. Discuss in how far the SystemC 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++/SystemC or similar).

Problem 1.4: Discrete-Event Semantics (30 points)

For each of the following code examples, what is the value of `myB` 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 SystemC 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)

```
void M::A(void)
{
  myB = 10;
};

void M::B(void)
{
  myB = 42;
};

SC_MODULE(M)
{
  int myB;

  void A(void);
  void B(void);

  SC_CTOR(M) {
    SC_THREAD(A);
    SC_THREAD(B);
  }
};
```

(b)

```
void M::A(void) {
  wait(42, SC_NS);
  myB = 10;
};

void M::B(void)
{
  myB = 42;
};

SC_MODULE(M)
{
  int myB;

  void A(void);
  void B(void);

  SC_CTOR(M) {
    SC_THREAD(A);
    SC_THREAD(B);
  }
};
```

(c)

```
void M::A(void) {
  myB = 10;
  wait(42, SC_NS);
};

void M::B(void) {
  wait(10, SC_NS);
  myB = 42;
};

SC_MODULE(M)
{
  int myB;

  void A(void);
  void B(void);

  SC_CTOR(M) {
    SC_THREAD(A);
    SC_THREAD(B);
  }
};
```

(d)

```
void M::A(void)
{
  myA = 10;
  e.notify();
  myA = 11;
  e.notify();
  wait(10, SC_NS);
};

void M::B(void)
{
  wait(e);
  myB = myA;
};

SC_MODULE(M)
{
  int myA;
  int myB;
  sc_event e;

  void A(void);
  void B(void);

  SC_CTOR(M) {
    SC_THREAD(A);
    SC_THREAD(B);
  }
};
```

(e)

```
void M::A(void)
{
  myA = 10;
  e.notify();
  wait(10, SC_NS);
  myA = 11;
  e.notify();
};

void M::B(void)
{
  wait(e);
  myB = myA;
};

SC_MODULE(M)
{
  int myA;
  int myB;
  sc_event e;

  void A(void);
  void B(void);

  SC_CTOR(M) {
    SC_THREAD(A);
    SC_THREAD(B);
  }
};
```

(f) What code has to be inserted at the beginning of M::B (line 12) in (e) to change the output of the program? What must *not* appear there for the program not to deadlock?

Embedded System Design and Modeling

ECE382N.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)

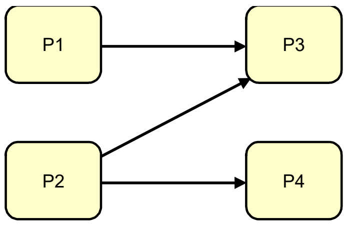
(a) Does Parks' KPN scheduling algorithm always find a bounded schedule if it exists? Why or why not?

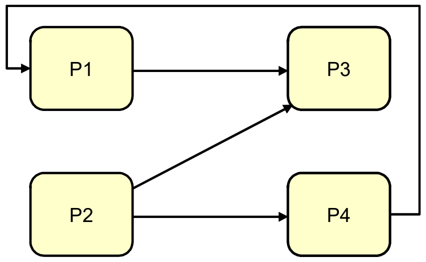
(b) Does Parks' algorithm always find a complete KPN schedule, if it exists? Why or why not?

(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)?

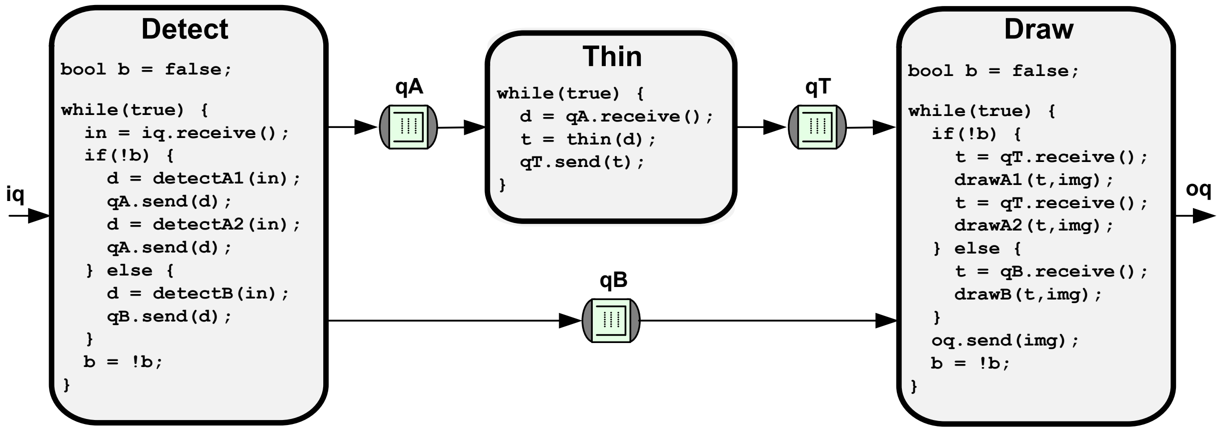
(b) In class, we mentioned that Parks' algorithm is not guaranteed to find a complete, bounded and non-terminating schedule even if one exists. Show an example of a KPN where such a schedule exists but Parks' algorithm fails to find it. Hint: in the KPN example presented in class, think about token patterns 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: KPN (20 points)

Given the following KPN model:



```
bool b = false;

while(true) {
  in = iq.receive();
  if(!b) {
    d = detectA1(in);
    qA.send(d);
    d = detectA2(in);
    qA.send(d);
  } else {
    d = detectB(in);
    qB.send(d);
  }
  b = !b;
}
```

```
while(true) {
  d = qA.receive();
  t = thin(d);
  qT.send(t);
}
```

```
bool b = false;

while(true) {
  if(!b) {
    t = qT.receive();
    drawA1(t,img);
    t = qT.receive();
    drawA2(t,img);
  } else {
    t = qB.receive();
    drawB(t,img);
  }
  oq.send(img);
  b = !b;
}
```

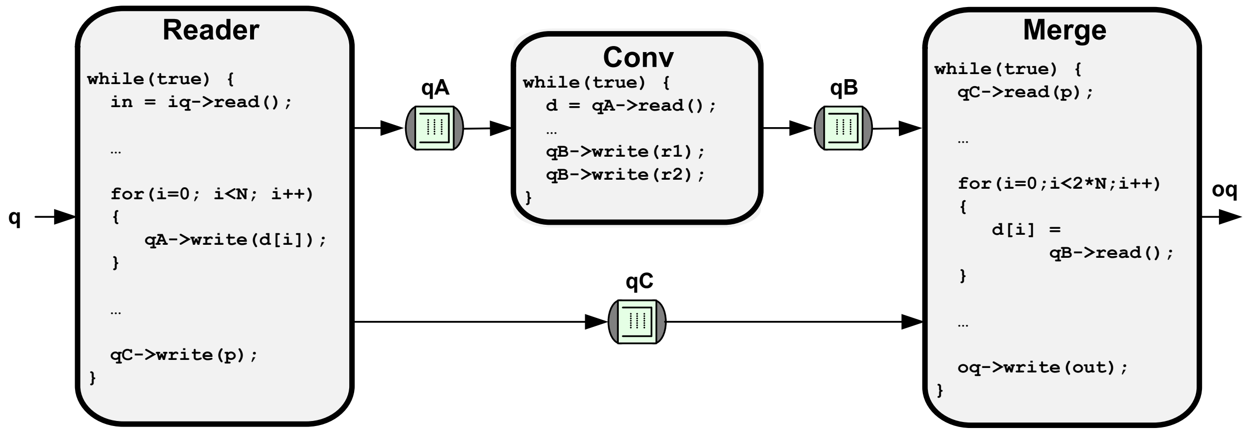
(a) Can this KPN be run in a complete, bounded and non-terminating manner? Show a data-driven schedule (as the sequence of *detectX()*, *thin()* and *drawX()* calls made) for processing of two *iq* tokens on a single processor with minimal number of context switches between processes. What is the buffer size needed for each queue?

(b) Show the schedule for processing of two *iq* items using Park's algorithm. What buffer size is needed? What is the minimum buffer size needed to run this model?

(c) Can this KPN be converted into an equivalent dynamic, Boolean, cyclo-static or synchronous dataflow model? If so, show the conversion into the most restrictive dataflow variant possible.

Problem 2.6: KPN (20 points)

Given the following KPN model:



Reader:

```
while(true) {
  in = iq->read();

  …

  for(i=0; i<N; i++)
  {
     qA->write(d[i]);
  }

  …

  qC->write(p);
}
```

Conv:

```
while(true) {
  d = qA->read();
  …
  qB->write(r1);
  qB->write(r2);
}
```

Merge:

```
while(true) {
  qC->read(p);

  …

  for(i=0;i<2*N;i++)
  {
     d[i] =
          qB->read();
  }

  …

  oq->write(out);
}
```

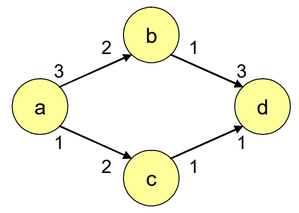
(a) Show one possible schedule when executing this KPN using Park's algorithm. Show the schedule in terms of queue reads and writes made by each process. Is Park's algorithm able to run this model in a complete, bounded and non-terminating manner? What is the buffer size needed for each queue?

(b) Can this KPN be executed in smaller memory (queue sizes) than what Park's requires? If not, why not? If yes, show the schedule and associated memory requirements.

(c) Can this KPN be converted into an equivalent synchronous dataflow model? If so, show the converted model. What impact does the conversion have on memory requirements (buffer sizes)?

Problem 2.7: 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.

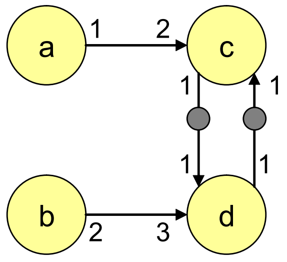
(d) Find the periodic schedule with the lowest token buffer usage. What is the minimum buffer usage?

(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 2.8: Synchronous Dataflow (15 points)

For the SDF graph on the right:

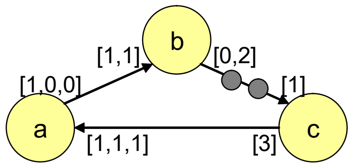


(a) Show that the graph is consistent, and that it has a valid schedule. What is the repetition vector of the graph? Give a minimal periodic static schedule if one exists.

(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 2.9: 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.

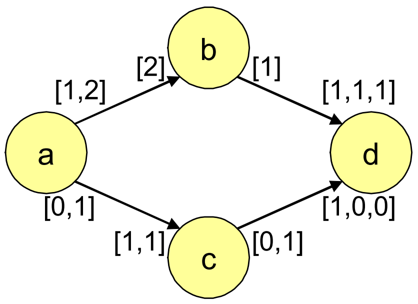
(c) List all possible minimal periodic static schedules.

(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 2.10: 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?

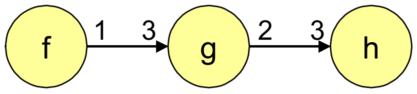
(b) Show the precedence graph for this CSDF.

(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?

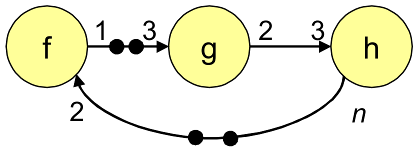
Problem 2.11: Synchronous Dataflow (SDF) (15 points)

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?

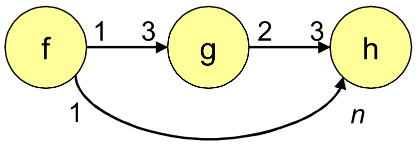
(a)



(b) *n* is an integer parameter.

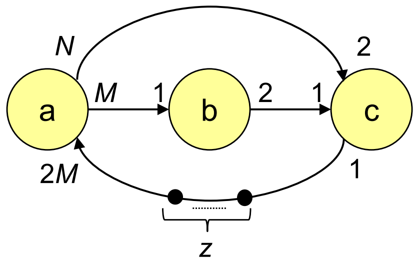


(c) *n* is an integer parameter.



Problem 2.12: Synchronous Dataflow (SDF) (15 points)

Given the following SDF graph, where *N*, *M* and *z* are integer parameters:



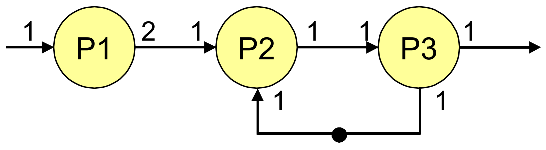
(a) Write down the balance equations and determine for which integer values of *N* and *M* the graph is consistent. Give the repetition vector for a minimal (least integer) schedule as a function of *N* and *M*.

(b) What is the minimal number of initial tokens *z* for which the graph is schedulable (as a function of *N* and *M*)? Set *N*, *M* and *z* to their minimal integer values that ensure consistency and schedulability, and write down a periodic single-processor schedule that minimizes buffer sizes. What is the minimal memory requirement to execute the graph?

(c) Now assume that $z = 0$. Is there an assignment of initial tokens to other channels that makes the graph schedulable? If so, what are the minimal values of initial tokens needed on other arcs? Set N and M to their minimal integer values that ensure consistency and find an assignment of initial tokens and a periodic single-processor schedule that minimize buffer sizes. What is the minimal memory requirement to execute the graph?

Problem 2.13: 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 2.14: 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:

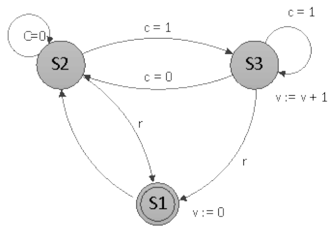
(a) Show an example of converting a hierarchical FSM into an equivalent plain FSM. Derive an expression for the complexity (number of states and number of transitions) of the equivalent plain FSM as a function of the complexity of the OR-composed FSMs.

(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.

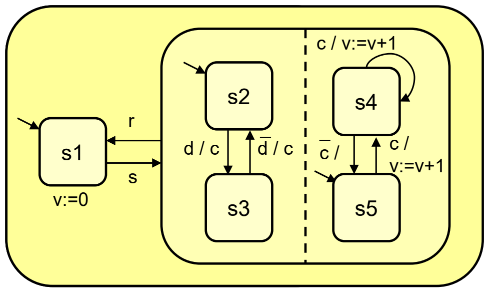
Problem 2.15: State Machines (15 points)

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.

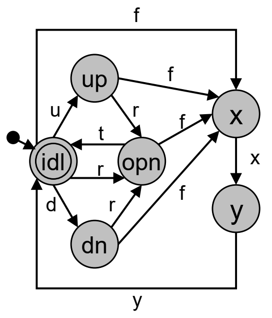
(a) Convert the extended FSM (EFSM)/FSMD on the right into an equivalent FSM. Can every FSMD be represented as a FSM? What can you say about the complexity (number of states) of the FSM as a function of the size and parameters of the original FSMD?



(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?



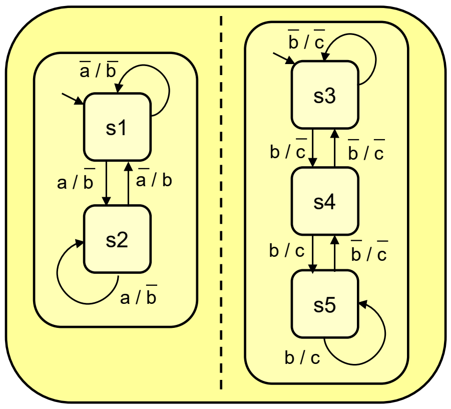
(c) Try to reduce the complexity of the FSM on the right by converting it into a HCFSM using hierarchy (OR states) and/or concurrency (AND states) wherever possible.



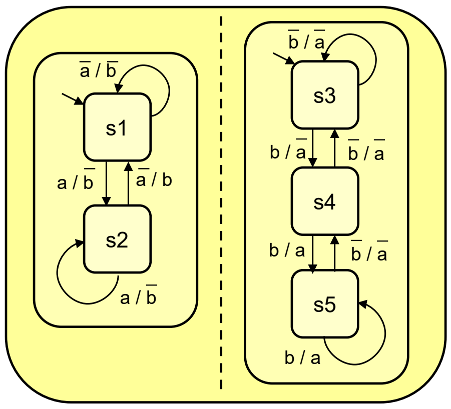
Problem 2.16: State Machine Models (20 points)

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.

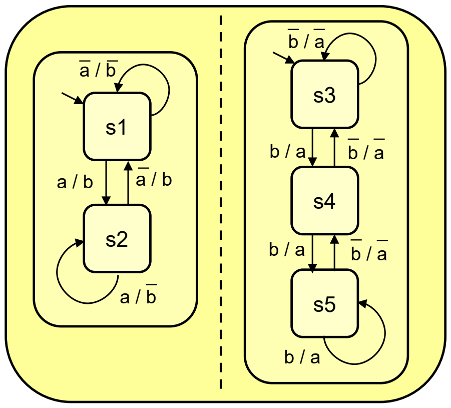
(a)



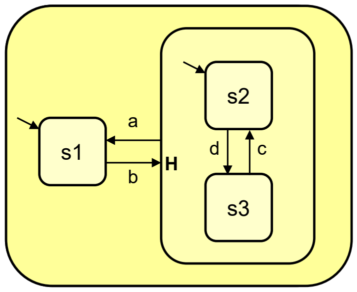
(b)



(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:



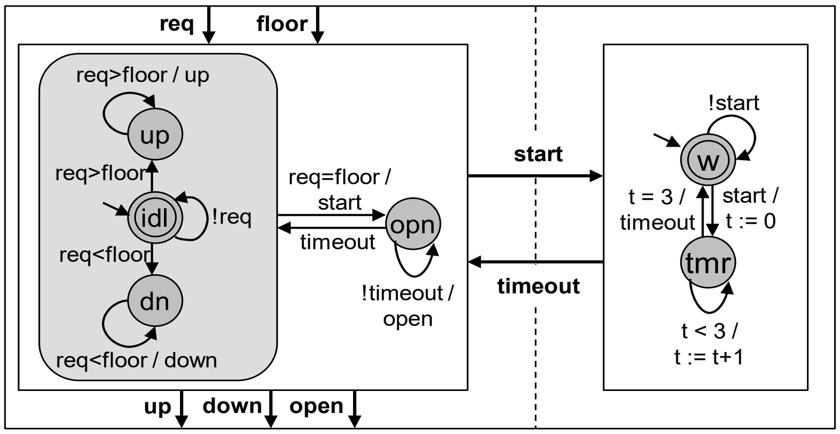
Problem 2.17: Synchronous/Reactive vs. State Machines (10 points)

In class we discussed that synchronous/reactive and HCFSM models are really based on the same (synchronous, i.e. lock-step and broadcast) semantics. As such, they can be translated into each other. Given the following synchronous/reactive Esterel program, draw its equivalent HCFSMD model.

```
module M:
input P;
output C: integer;  %// valued signal
signal I in  %// local signal
  [
    loop await P; emit I; await not P; emit I; end
  ||
    var v:=0: integer in  %// local variable, init to zero
      loop await I; v:=v+1; emit C(v); end
    end var
  ];
 end signal
end module
```

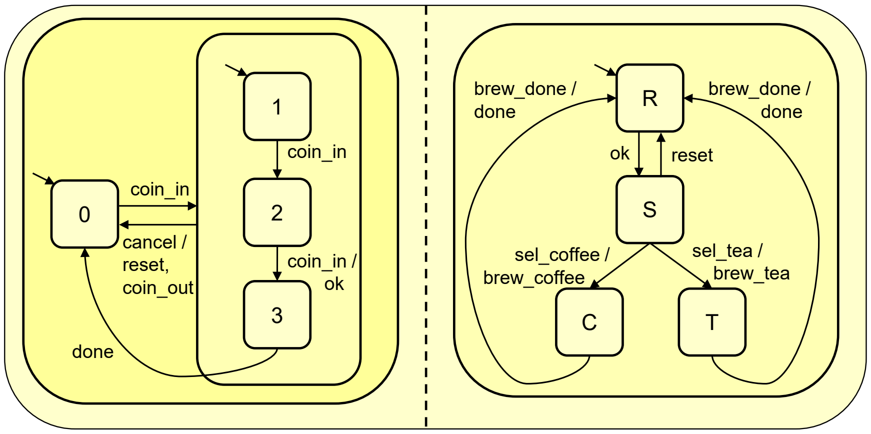
Problem 2.18: State Machines (10 points)

Convert the following HCFSMD model of an elevator controller consisting of two concurrent, communicating state machines into an equivalent single, flat Mealy FSM *without* data (i.e. an FSM, *not* an FSMD). The HCFSMD has two external inputs (requested and current floor), three external outputs (up/down motor control and door open signals), and two internal signals (start timer and timer timeout). Unless specified otherwise, default value for all signals is absent/0.



Problem 2.19: HCFSM (20 points)

Given is the following HCFSM model of a simple vending machine. The machine has 5 external inputs (a *coin_in* slot, a *cancel*, *sel_tea* and *sel_coffee* button, and a *brew_done* signal), 3 external outputs (*brew_tea* and *brew_coffee* signals to trigger the brew unit operation, and a *coin_out* signal to release inserted coins), and 3 internal signals for communication between state machines (*ok*, *reset*, *done*). Unless specified otherwise, signals are by default absent and states have implicit self-transitions.



(a) Demonstrate the operation of the vending machine for a customer ordering tea. Show the sequence of events and state transitions.

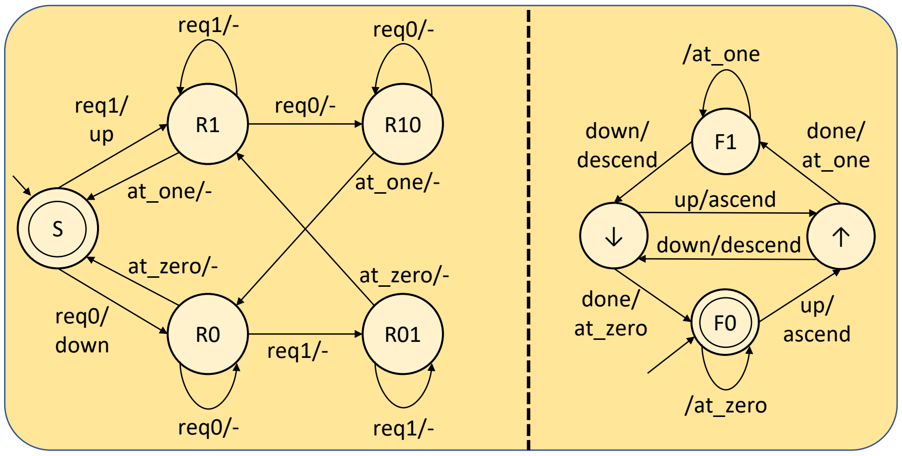
(b) Do you see any way to further simplify and reduce complexity (e.g., number of states or number of transitions) of the model using any of the state machine concepts we discussed in class? If so, just sketch where and how the model could be changed, you don't need to show the complete simplified state machine.

(c) Convert the HCFSM into an equivalent single, flat FSM.

(d) The state machine has a bug that allows the user to cheat. Show a trace of input events and transitions that demonstrates the bug. Show how the original HCFSM and the flattened FSM need to be modified to fix the bug (you can indicate changes directly in the graphs above).

Problem 2.20: HCFSM (20 points)

Given in the following is the HCFSM model of an elevator control system for a two-story building. The machine has 3 external inputs: *req0* (for requesting elevator from floor #0), *req1* (for requesting elevator from floor #1), *done* (notification of mechanical system), 2 external outputs: *ascend* and *descend* (commands for mechanical system), and 4 internal signals for communication between state machines (*up*, *down*, *at_one*, *at_zero*). Unless specified otherwise, signals are by default absent and states have implicit self-transitions.



(a) Demonstrate the operation of the elevator for answering a request from floor #1 and going from floor #0 to floor #1. You can assume that the elevator starts at floor #0 (corresponding to start states *S* and *F0*). Show the sequence of events and state transitions.

(b) Convert the HCFSM into an equivalent single, flat FSM.

(c) The state machine has a bug that can make elevator get stuck at a floor. Show a trace of input events and transitions that demonstrates the bug. Show how the original HCFSM and the flattened FSM need to be modified to fix the bug (you can indicate changes directly in the graphs above).

Embedded System Design and Modeling

ECE382N.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)

(a) 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?

(b) 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.

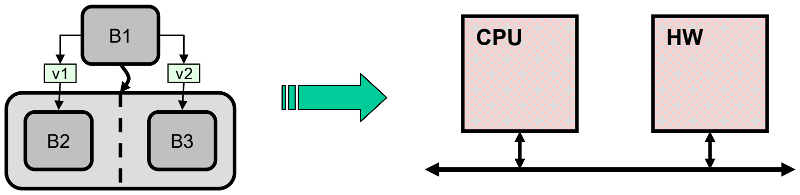
Problem 3.4: Amdahl's Law (10 points)

(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 2.8, 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.

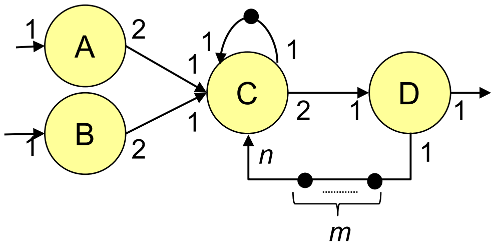
(b) Apply a list scheduling algorithm that uses the level (longest distance to the sink, i.e. critical path length) of a node in the graph as priority function. Show the step-by-step operation of the algorithm (state of the priority-sorted ready queue and mapping decisions made in each step), as well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

Problem 3.8: SDF Mapping (35 points)

Given the SDF graph below, 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).



(d) Determine whether a valid schedule exists that can be executed repeatedly and indefinitely in bounded buffer memory. Write down the balance equations, determine integer values for n for which the graph is consistent and give the repetition vector for a minimal (least integer) schedule. For what number of tokens m on the D→C arc is the graph schedulable? Fix n to its minimal value and m=2n for all following questions.

(e) 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.

(f) 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?

(g) 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?

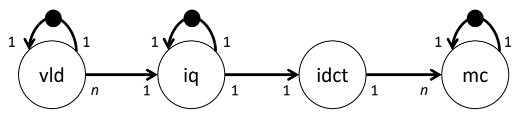
(h) Assume that you can schedule the graph in a pipelined fashion, where one or more consecutive iterations overlap, i.e. a new iteration can begin while previous iterations are still running. Show the schedules with minimal latency and maximal throughput on 2 processors. You can assume that different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

(i) 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.

(j) 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).

Problem 3.9: Multiprocessor Mapping (25 points)

Given the following dataflow graph of a H.263 decoder, where *n* is the number of macro-blocks in each image frame. Explore mappings of this graph onto a heterogeneous smartphone platform consisting of a quad-core ARM processor and a GPU with the following actor execution times (N/A means that the actor can not be mapped onto the respective processor type):



	ARM	GPU
vld	25	10
iq	1	N/A
idct	1	N/A
mc	10	5

(a) What is the repetition vector for this SDF graph? Convert the graph into an equivalent homogeneous SDF (HSDF) model according to its repetition vector, i.e. show the precedence graph for this example.

(b) Set *n* = 3 and apply a list scheduling algorithm to map and schedule one iteration of the graph onto the quad-core ARM processor without utilizing the GPU. Use the maximum distance to the sink (i.e. length of the longest/critical path with the maximum total execution time to reach the sink) as priority function for the list scheduler. Show the step-by-step operation of the algorithm (time, state of the ready list with priorities and mapping decisions made in each step), as well as the final schedule. What is the latency of your schedule and how many ARM cores do you need?

(c) Schedule the graph (with *n* = 3) in a pipelined fashion, where one or more consecutive iterations can overlap. What is the highest throughput that can be achieved by running on the quad-core ARM processor only? Show a schedule that achieves this throughput while minimizing latency. What is the latency of your schedule and how many cores do you need?

(d) Now schedule the graph (with *n* = 3) utilizing both the quad-core ARM and the GPU. Show a schedule that achieves minimal latency, and a schedule that achieves maximal throughput with minimized latency. What is the latency and throughput, and how many cores do you need in each case?