Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Institut für Technische Informatik
und Kommunikationsnetze
Computer Engineering and Networks Laboratory

Exam Autumn 2016

Hardware/Software Codesign

Prof. L. Thiele

Note:

The given solution is only a proposal. For correctness, completeness, or understandability, no responsibility is taken.

Task 1 : Models of Computation (maximal 43 points)

1.1: StateCharts (maximal 13 points)

Consider the StateChart as shown in Figure 1.

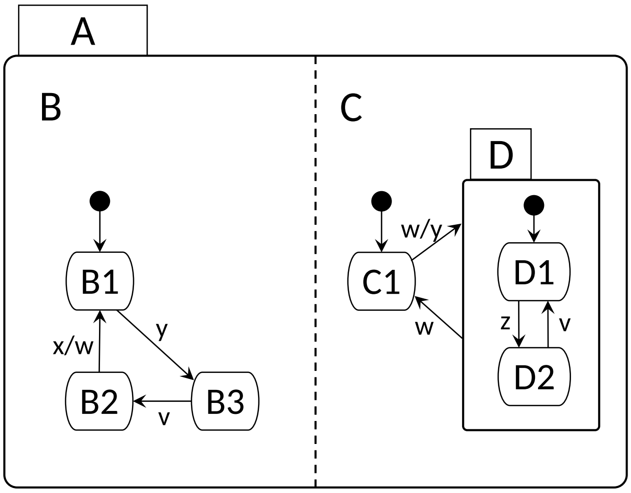

Figure 1: StateChart

(a) (5 points) Draw the state space of the StateChart as a tree, which shows the hierarchy of states and denote the state types (basic states, sequential states, and parallel states).

Sample solution: See Figure 2

(b) (8 points) Consider the following sequence of external events: x, w, y, y, z, v, x. Determine the sequence of states, the automaton in Figure 1 passes through, starting from the initial state.

Sample solution: See Table 1

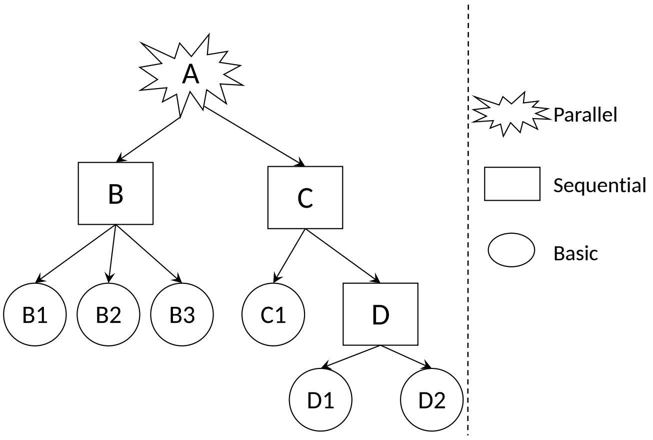

Figure 2: Solution of 1.1 (a)

event	state
initial	B_1C_1
x	B_1C_1
w	B_3D_1
y	B_3D_1
y	B_3D_1
z	B_3D_2
v	B_2D_1
x	B_1C_1

Table 1: Solution of 1.1 (b)

1.2: Dataflow Languages (maximal 21 points)

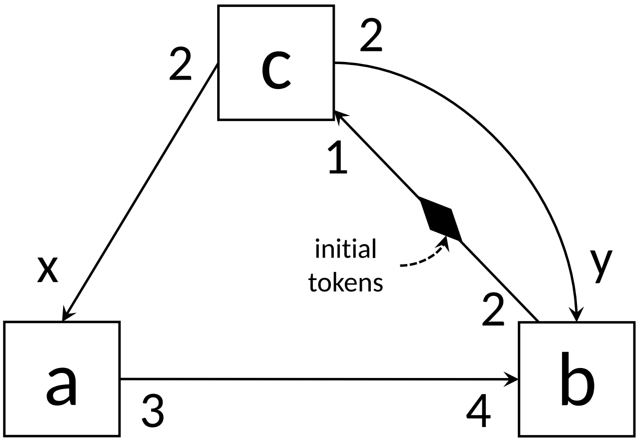

Figure 3: SDF graph

Figure 3 shows a system modeled by a SDF graph with processes $\{a, b, c\}$. The numbers attached to the heads and tails of edges represent the number of tokens consumed and produced respectively, at each firing of the corresponding process.

(a) (3 points) Write down the topology matrix of the system.

Sample solution:

$$M = \begin{bmatrix} 3 & -4 & 0 \\ -x & 0 & 2 \\ 0 & 2 & -1 \\ 0 & -y & 2 \end{bmatrix}$$

(b) (8 points) Determine the parameters x and y such that the system has a periodic schedule and note the relative execution rates of the processes.

Sample solution: A connected SDF graph with n actors has a periodic schedule *iff* its topology matrix M has rank $n - 1$. For this exam question, we get rank(M)=2. This means that the columns of the matrix are linearly dependent:

$$\alpha \times \begin{bmatrix} 3 \\ -x \\ 0 \\ 0 \end{bmatrix} + \beta \times \begin{bmatrix} -4 \\ 0 \\ 2 \\ -y \end{bmatrix} = \begin{bmatrix} 0 \\ 2 \\ -1 \\ 2 \end{bmatrix}$$

We get $\alpha = -\frac{2}{3}$, $\beta = -\frac{1}{2}$, $x = 3$, $y = 4$. The 1st and 2nd columns are still independent, rank(M)=2. Instantiate x and y in M and solve the "state equation":

$$M \times \begin{bmatrix} a \\ b \\ c \end{bmatrix} = 0$$

We get $a = \frac{4}{3}b$ and $c = 2b$. Therefore the relative firing rates are represented as

$$\begin{bmatrix} a \\ b \\ c \end{bmatrix} = \begin{bmatrix} 4 \\ 3 \\ 6 \end{bmatrix}$$

(c) (10 points) Determine a periodic schedule that requires a minimum number of initial tokens. **Note:** Only edge bc has initial tokens.

Sample solution:

- 1 token on edge BC. Schedule: $c, \text{blocked}$
- 2 tokens on edge BC. Schedule: $2c, a, \text{blocked}$
- 3 tokens on edge BC. Schedule: $3c, 2a, b, 2c, a, b, c, a, b$. At the end of this schedule, edge BC has 3 tokens and all other edges have 0 tokens (initial condition). Therefore, this periodic schedule repeats.

1.3: Parks' Algorithm (maximal 9 points)

Apply Parks' algorithm to the SDF graph in Figure 4 and determine the required buffer sizes. **Note:** $\mathcal{S}_{ab}$ and $\mathcal{S}_{bc}$ denote the buffer sizes of channels ab and bc, respectively.

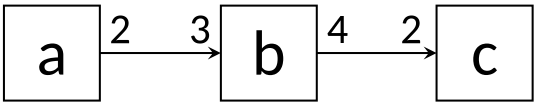

Figure 4: SDF graph with 3 processes $\{a, b, c\}$ – numbers on edges have the same semantics as shown in question 1.2.

Sample solution: After the firing sequence shown in the Table 2, the buffer states reset to contain zero tokens, and the schedule repeats. Therefore, the buffer size is 4 for each of the channels. Note: multiple sequences exist, all leading to same requirements on minimum buffer capacities.

fired process	$\mathcal{S}_{ab}$	$\mathcal{S}_{bc}$
– (initial)	1	1
a	2	1
a	4	1
b	4	4
c	4	4
c	4	4
a	4	4
b	4	4
c	4	4
c	4	4

Table 2: Solution of 1.3

Task 2 : Performance Analysis (maximal 39 points)

2.1: WCET Analysis (maximal 21 points)

Determine the WCET of the following program which is written in pseudo code.

```
s=0;

WHILE (k < 5) do {

  if (f)
    j = j + 1;
  else {
    j = 0;
    f = TRUE;
  }

  k = k + 1;
  s = s + k;
}
```

The underlying machine executing the program is specified as follows:

- the processor has no pipeline.
- the processor has no registers, i.e. all variables are stored in memory.
- in an assignment, predicate evaluation, or arithmetic operation, variables are accessed in order from right to left e.g.

  `a = b + c`

  variables are accessed in the order c, b, a.
- consider only the data cache.
- reads and writes from/to a variable are treated the same way. If the required data is in the cache, there is a cache HIT and it takes **1 cycle** to read/write the data. If the data is not in the cache, there is a cache MISS and it takes **40 cycles** to read/write the data.

- execution model is very simplified. No temporary results are considered. Execution times of all processor operations are 1 cycle. In addition, the time for loading and storing a variable needs to be taken into account.
- the data cache is fully associative. It can store 4 blocks of data. Each cache block can contain exactly one variable.
- the cache uses LRU replacement policy.

Initial conditions: $s, j \in (-\infty, \infty)$, $f \in [0, 1]$, and $k \in [0, +\infty)$ and cache state is $(-, -, -, -)$, i.e. all blocks are empty.

(a) (5 points) Determine the basic blocks of the program.

Sample solution: *See Figure 5.*

(b) (10 points) Use cache MUST analysis to determine the worst-case execution time for each basic block. **Do not** unroll the loop.

Sample solution: *Using cache MUST analysis to determine the WCET of each basic block.*

```
H for hit, M for miss,
ci is WCET in cycles for the i-th basic block
Initial cache state:  ({},{},{},{})

B1:
  s = 0
    M, ({s},{},{},{})
  c1: 40

First pass through the loop
with initial cache state: ({s},{},{},{})

B2:
  while(k<5)
    M, ({k},{s},{},{})
  c2: 41

B3:
  if(f)
```

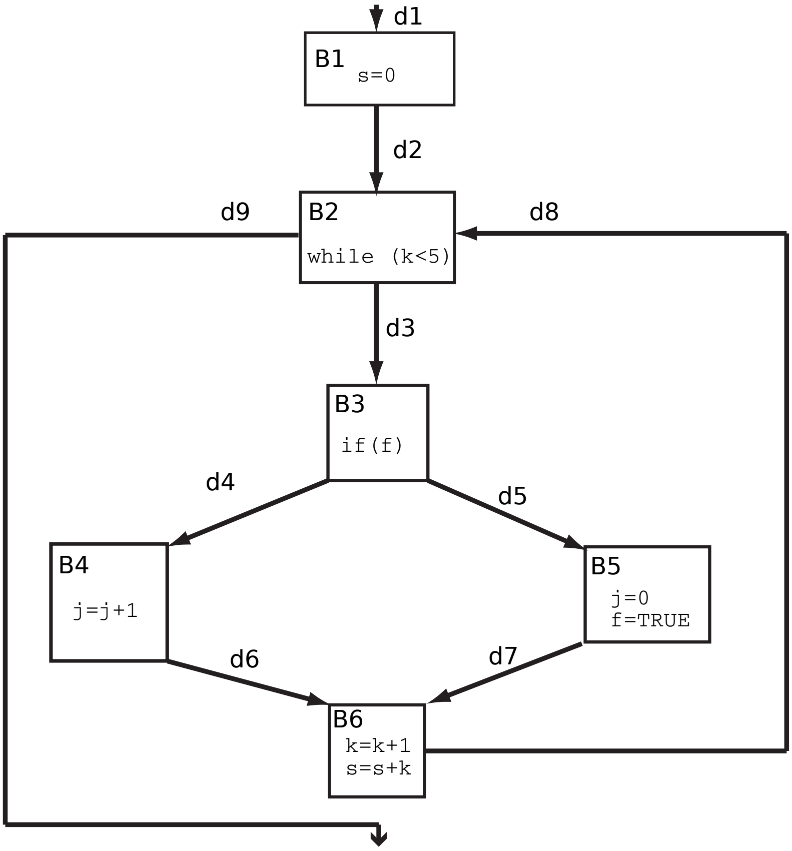

Figure 5: Basic blocks of the program

```
    M, ({f},{k},{s},{})
  c3: 41

B4:
  j = j+1
    M, ({j},{f},{k},{s})
    H, ({j},{f},{k},{s})
  c5 = 42

B5:
  j=0
    M, ({j},{f},{k},{s})
  f = TRUE
```

```
    H, ({f},{j},{k},{s})
  c4: 41

Join of B4 and B5:  ({},{f,j},{k},{s})

B6:
  k = k + 1
    H, ({k},{},{f,j},{s})
    H, ({k},{},{f,j},{s})
  s = s + k
    H, ({k},{},{f,j},{s})
    H, ({s},{k},{},{f,j})
    H, ({s},{k},{},{f,j})
  c6 = 7

Join of B6 and B1: ({s},{},{},{})
```

`({s},{},{},{})` is the static abstract cache state. It is not necessary to analyze more loop iterations, since the cache state is already stable.

In this example, we can see that variables `f,j,k` are always pushed out of the abstract cache state (at the join of B6 with B1) even though they were always accessed in the loop. This will make the analysis conclude wrongly that there are always cache misses for all subsequent iterations of the `while(k < 5)` loop. These are therefore pessimistic bounds, which cannot be improved without loop unrolling.

(c) (6 points) Write down the structural constraints and the ILP to determine the WCET of the program. Given the initial value intervals of $s, j \in (-\infty, +\infty)$, $f \in [0, 1]$, and $k \in [0, +\infty)$, can you specify any additional constraints?

Sample solution: *Flow variables d_i are defined in Figure 5.*

Structural constraints:

$$
\begin{aligned}
d_1 = d_2 &= x_1 \\
d_2 + d_8 = d_3 + d_9 &= x_2 \\
d_3 = d_4 + d_5 &= x_3 \\
d_4 = d_6 &= x_4 \\
d_5 = d_7 &= x_5 \\
d_6 + d_7 = d_8 &= x_6
\end{aligned}
$$

Assume program is executed once: $d_1 = 1$.

Additional constraints:

- *Loop bounds: considering the different paths in the program and variable values, loop is executed at most 5 times:* $x_3 \leq 5 * x_1$.
- *Program is executed once:* $x_1 = 1$

Objective function to maximize:

$$WCET = \sum_{i=1}^{6} c_i * x_i$$

2.2: Modular Performance Analysis (maximal 18 points)

A bus has to transmit data of two independent packet streams. The arbitration policy used is fixed priority, i.e. the packets of the first stream have the high priority and the bus communicates data from the second stream only if there is no waiting data from the first stream. Data arrives in packets but the transmission of data can be interrupted and resumed any time (preemptive arbitration).

The input streams and the bus are characterized as follows:

- *Stream 1:* Periodic stream with jitter, packet size 1KByte, period 2ms, jitter 1.5ms.
- *Stream 2:* Periodic stream, packet size 1KByte, period 2ms.
- *Bus:* Bandwidth 1MByte/s.

(a) (4 points) Draw the upper arrival curve for stream 1 and the service curve for the bus using Figure 6.

Sample solution: To calculate the upper arrival curve, the jitter needs to be taken into account. The smallest time difference between any two packets is 0.5 ms. After that, one full period is necessary to receive another packet. The bus service curve is a single line with slope 1MByte/s, since it can transmit one packet per ms. The arrival curve from stream 1 can be seen in red in Figure 6. The service curve of the bus is in blue.

(b) (2 points) Determine the maximum data delay (in ms) and maximum capacity of the buffer (in KBytes) for stream 1. Justify your answer.

Sample solution: To calculate the the maximum delay, it is necessary to calculate the maximum horizontal distance between the upper arrival curve and lower service curve. From Figure 6, it can be seen that this happens at 2 ms when two packets from stream 1 have arrived. Hence, the maximum delay is 1.5 ms. To calculate the maximum capacity, it is necessary to calculate the maximum vertical distance between the upper arrival curve and lower service curve. From Figure 6, it can be seen that this occurs at $\Delta = 0.5$ ms, where the distance is 1.5. Hence, the minimum buffer size is 1.5 KBytes. **Alternate solution:** if is it assumed (and stated) that buffer sizes can only be integer and must be rounded up, then 2 packets need to be buffered, requiring 2 KBytes.

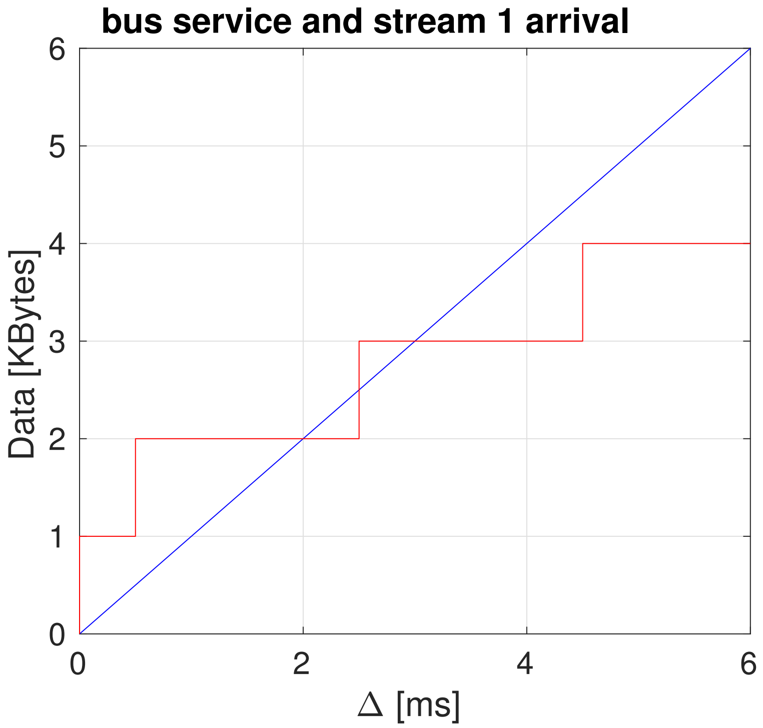

Figure 6: Arrival curves for stream 1 (red) and bus service (blue)

(c) (10 points) Draw the upper arrival curve for stream 2 and the lower service curve that is left over for stream 2 after communicating data of stream 1. Use Figure 7.

Sample solution: To calculate the upper arrival curve, only the period needs to be taken into account. This produces the ladder shape, which increases after every period. Since stream 2 has the lower priority, it only receives the service remaining from stream 1. To determine the lower curve of this remaining service, one can apply the following formula: $\sup_{0 \leq \lambda \leq \Delta} (\beta^l(\lambda) - \alpha^u(\lambda))$. To calculate this manually, we can transform Δ to the time domain, and determine the bus' idle times. From Figure 6, it can be seen that, in the worst case, the bus can be busy during the first 2 ms processing the first two packets. For $t \in [2, 2.5]$, the bus is idle since no packets from stream 1 arrive. Therefore, in [2, 2.5], the service curve for Stream 2 increases with a slope of 1MByte/s. After 2.5 ms, the bus exhibits periodic behavior: busy for 1 ms servicing stream 1, idle for 1 ms. Therefore, the service curve for stream 2 follows the following periodic pattern: slope of 0 for 1 ms, slope of 1MByte/s for 1 ms. The arrival curve of stream 2 can be seen in red in Figure

7. The service curve for stream 2 is in blue.

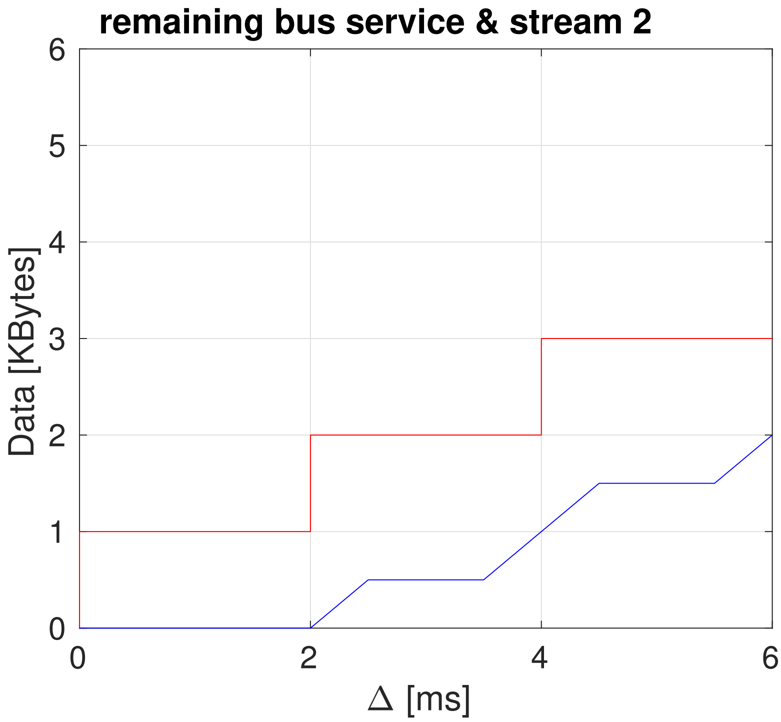

Figure 7: Arrival curve for stream 2 (red) and the bus service remaining from stream 1 (blue)

(d) (2 points) Determine the maximum data delay (in ms) and maximum capacity of the buffer (in KBytes) for stream 2. Justify your answer.

Sample solution: To calculate the maximum delay, it is necessary to calculate the maximum horizontal distance between the upper arrival curve and lower service curve. From Figure 7, it can be seen that this happens when there is a new arrival (every period). Hence, the maximum delay is 4 ms. To calculate the maximum capacity, it is necessary to calculate the maximum vertical distance between the upper arrival curve and lower service curve. From Figure 7, it can be seen that this occurs at $\Delta = 2, 4, \ldots$, where the distance is 2. Consequently, 2 packets need to be buffered, requiring 2 KBytes.

Task 3 : Optimization, Mapping and Partitioning, and Design Space Exploration (maximal 38 points)

3.1: Multi-Criteria Optimization (maximal 14 points)

Figure 8 shows a 2-dimensional design space with nine design points. Both objectives, the execution time T, and the power dissipation P, are to be minimized.

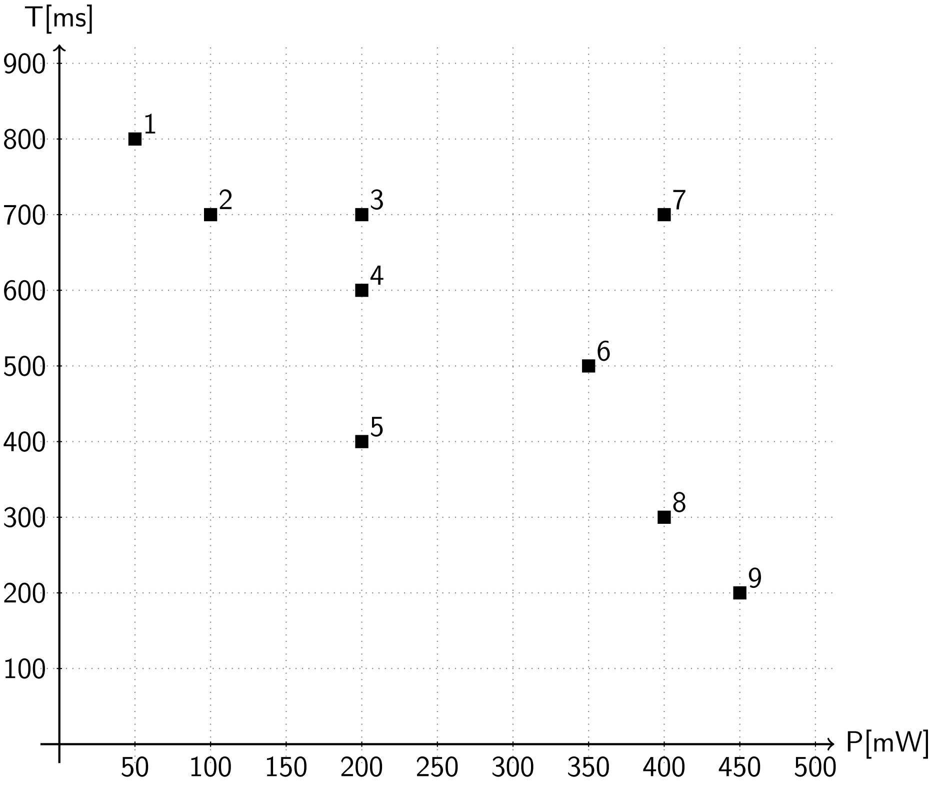

Figure 8: Design space

(a) (2 points) Indicate the Pareto points in Figure 8.

Sample solution: These are points 1, 2, 5, 8, and 9 (marked red in figure 9).

(b) (3 points) Given the reference point $R(500\text{mW}, 900\text{ms})$, compute the hypervolume of the given nine design points.

Sample solution: The hypervolume is obtained from the figure, and is 38 (shaded gray in figure 9).

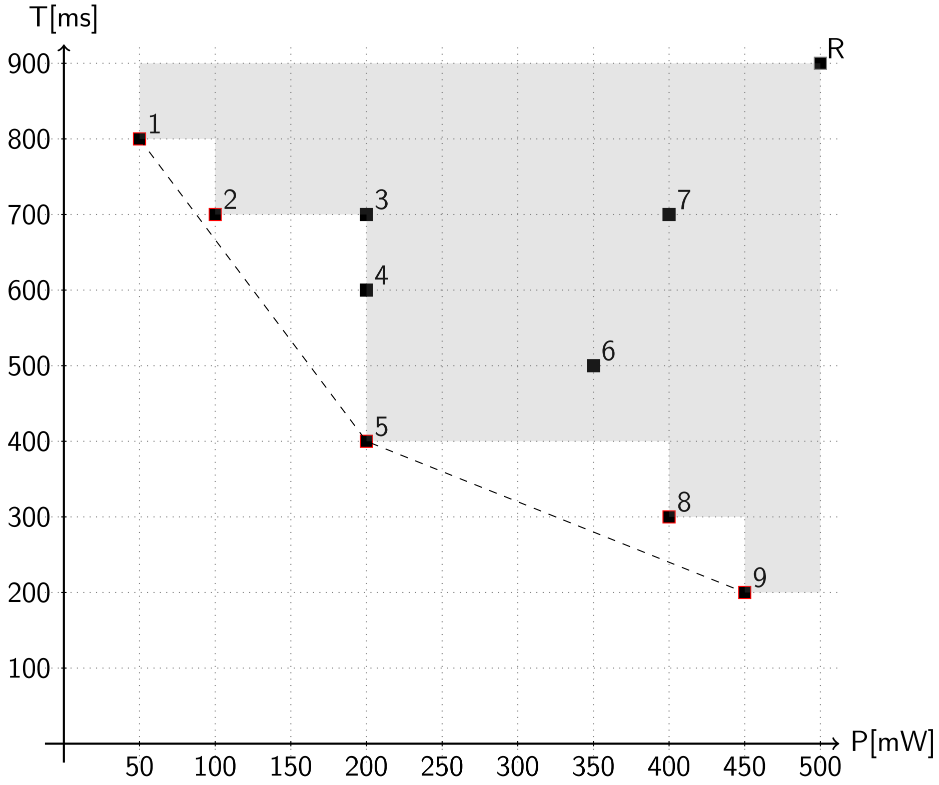

Figure 9: Design space

(c) (3 points) Given the above mentioned reference point, find the subset of three points with the *largest* hypervolume (environmental selection).

Sample solution: Removing design point 1, and either 8 or 9, will cause the hypervolume to be the largest (36). Therefore, the subsets $\{2, 5, 8\}$ or $\{2, 5, 9\}$ give the largest hypervolume. Note: just one solution needed.

(d) (6 points) A single-objective optimization method is used which minimizes a weighted cost function: $cost(d) = k_1 \cdot T(d) + k_2 \cdot P(d)$, where d denotes a design point and $k_1 + k_2 = 1$.

(a) (5 points) Which design point is returned for $(k_1, k_2) = (0.8, 0.2)$ and what is its cost?

Sample solution: Point 9 is returned, with cost 250.

(b) (1 point) Which of the Pareto points in Figure 8 will not be found by this method, irrespective of the weights chosen?

Sample solution: Points 2 and 8 will not be found by this method, irrespective of the weights chosen. This is because points 2 and 8 are not convex (as illustrated by the dashed line in Figure 9)

3.2: Design Space Exploration (maximal 24 points)

Figure 10 shows a specification at the system level consisting of a data flow graph and an architecture template. Table 3 presents the available components with cost and execution times, as well as the time taken for bus communications. We have one instance of each of the components available. Communication between tasks bound to the same component does not require any communication time.

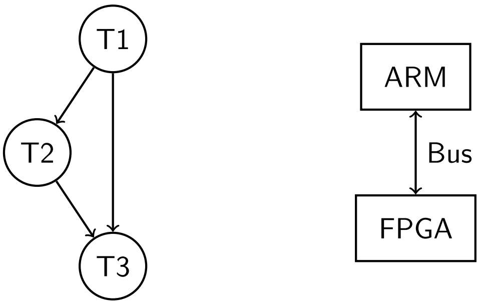

Figure 10: System-level specification

component	*cost* [CHF]	*execution time* $[ms]$		
		T1	T2	T3
ARM	50,-	10	5	8
FPGA	100,-	8	2	4

component	*cost* [CHF]	*communication time* $[ms]$		
		T1→T2	T1→T3	T2→T3
Bus	10,-	5	3	3

Table 3: Components with cost and execution times for tasks and bus communications

(a) (3 points) Provide tight upper bounds, without considering the task precedence constrains, for the number of distinct bindings and schedules.

Sample solution: Bindings $2^3 = 8$, schedules $3! = 6$.

(b) (3 points) Construct the specification graph. Your graph should include the problem graph with communication nodes, the architecture graph, and all possible bindings.

Sample solution: See Figure 11.

(c) (14 points) List all design points (allocation, binding, schedule) together with their total execution time and cost. Insert the design points into the time-cost diagram in Figure 12. Identify and mark the Pareto points.

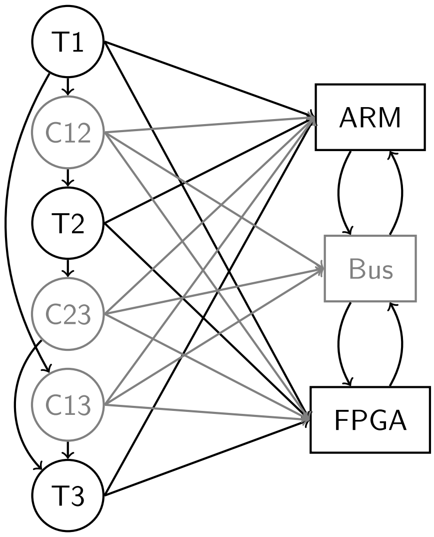

Figure 11: Specification graph

Sample solution: The following solution assumes that communications between different tasks can happen in parallel. Schedule is not listed in Table 4 because it is always T1→T2→T3. The Pareto points in Figure 12 are marked in black.

	#1	#2	#3	#4	#5	#6	#7	#8
T1	ARM	FPGA	ARM	FPGA	ARM	FPGA	ARM	FPGA
T2	ARM	ARM	FPGA	FPGA	ARM	ARM	FPGA	FPGA
T3	ARM	ARM	ARM	ARM	FPGA	FPGA	FPGA	FPGA
exec.	23	26	28	21	22	25	21	14
CHF	50	160	160	160	160	160	160	100

Table 4: Design points

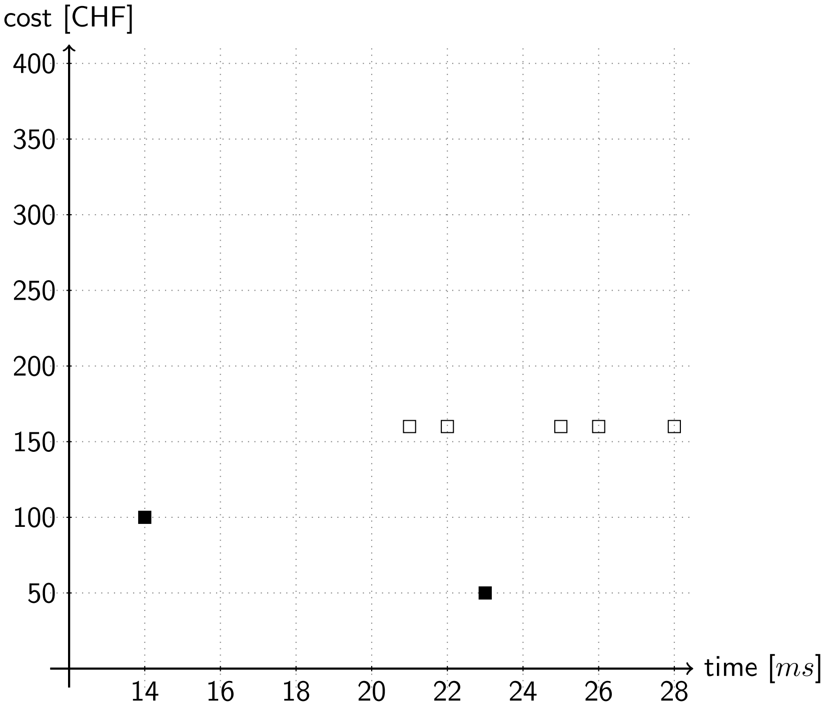

Figure 12: Time-cost diagram

(d) (2 points) The design space exploration is done with an evolutionary algorithm. Present a possible encoding for the allocation and binding.

Sample solution: We can use $a_1a_2b_1b_2b_3$, where a_1 denotes the usage of the ARM, a_2 the usage of the FPGA, and b_i means that task Ti is binded to the ARM (0) or the FPGA (1).

(e) (2 points) Evaluate the redundancy of the encoding chosen in the task before by comparing the sizes of the search and solution space.

Sample solution: The search space of the $a_1a_2b_1b_2b_3$ encoding has 32 points, while the design space has 8 points. The redundancy is thus four.