Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Institut für Technische Informatik
und Kommunikationsnetze
Computer Engineering and Networks Laboratory

Exam Spring 2015

Embedded Systems

Prof. L. Thiele

Note:

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

Task 1 : Real-Time Scheduling (maximal 40 points)

1.1: Periodic Task Scheduling (maximal 20 points)

A *harmonic* periodic task set is a periodic task set where all periods of the tasks are pairwise multiples or divisors of each other. Table 1 shows an example of a harmonic periodic task set.

	Computation Time	Period = Deadline
Task 1	1	2
Task 2	1	4
Task 3	2	8

Table 1: Harmonic periodic task set.

(a) (5 points) Schedule all tasks in Table 1 using Rate Monotonic (RM) scheduling. Draw the RM schedule in Figure 1. Do all tasks meet their deadlines?

Sample solution:
The priorities are Task 1, then Task 2, then Task 3. All deadlines are met.

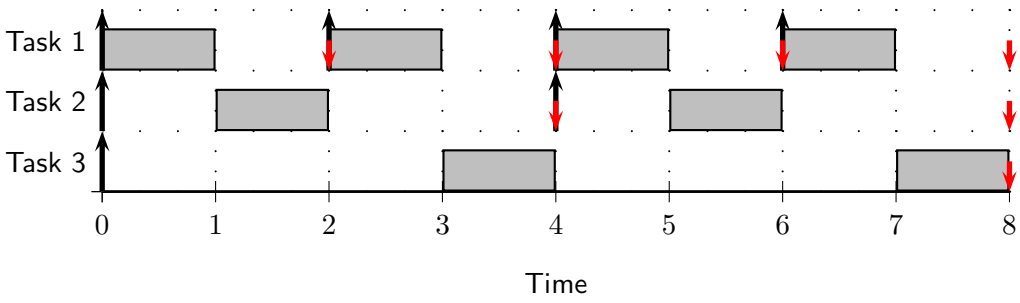


Figure 1: RM Schedule.

(b) (5 points) Schedule all tasks in Table 1 using Earliest Deadline First (EDF). In case of equal deadlines, a task with lower *index* has higher priority. Draw the EDF schedule in Figure 2. Is the EDF schedule the same as the RM schedule?

Sample solution:
RM and EDF Schedules are the same.

□

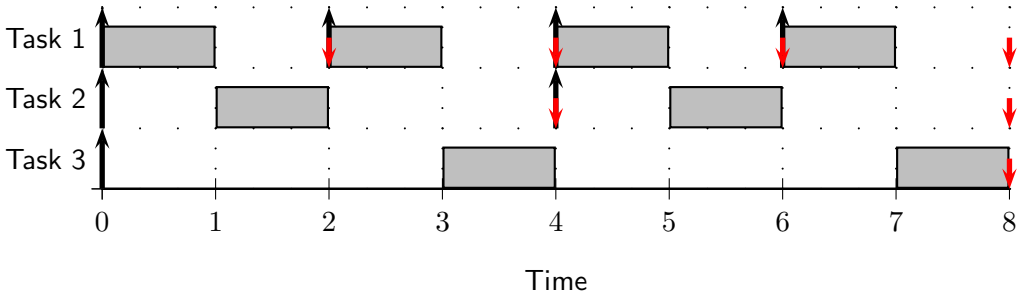


Figure 2: EDF Schedule.

(c) (10 points) Let C_i and T_i denote the computation time and the period of the periodic task i, respectively. Assume that the relative deadline of each task is equal to its period. Prove the following statement:

If a periodic task set with n tasks is *harmonic* (i.e., for any two tasks i and j, if $T_i \geq T_j$ then $\frac{T_i}{T_j}$ is a positive integer) and it holds that $\sum_{1 \leq i \leq n} \frac{C_i}{T_i} \leq 1$, then all tasks of the task set meet their deadline using the RM scheduling policy.

(Hint: Start by using the necessary and sufficient schedulability test of the RM scheduling algorithm for the lowest priority task)

Sample solution:
Assume n tasks ordered in descending order of priority/rate. Sufficient RM feasibility condition for lowest priority task is:

$$\sum_{1 \leq i \leq n-1} \left(C_i \cdot \left\lceil \frac{T_n}{T_i} \right\rceil \right) + C_n \leq T_n$$

Since tasks are harmonic $\left\lceil \frac{T_n}{T_i} \right\rceil = \frac{T_n}{T_i}$. Therefore:

$$\sum_{1 \leq i \leq n-1} \left(C_i \cdot \frac{T_n}{T_i \cdot T_n} \right) + \frac{C_n}{T_n} \leq 1 \implies \sum_{1 \leq i \leq n} \frac{C_i}{T_i} \leq 1$$

For task j, the necessary schedulability condition is:

$$\sum_{1 \leq i \leq j-1} \left(C_i \cdot \frac{T_j}{T_i \cdot T_n} \right) + \frac{C_j}{T_j} \leq 1 \implies \sum_{1 \leq i \leq j} \frac{C_i}{T_i} \leq 1$$

Furthermore:

$$\sum_{1 \le i \le j} \frac{C_i}{T_i} \le \sum_{1 \le i \le n} \frac{C_i}{T_i} \qquad \forall j \le n$$

Therefore, $\sum_{1 \le i \le n} \frac{C_i}{T_i} \le 1$ is a sufficient condition for RM feasibility of a harmonic periodic task set.

1.2: Aperiodic Task Scheduling (maximal 7 points)

An application consists of periodic tasks and three aperiodic tasks. The total utilization of periodic tasks is 0.6. Parameters of aperiodic tasks are given in Table 2.

	Arrival Time	Computation Time
J_1	57	10.5
J_2	60	37.2
J_3	180	65.4

Table 2: Aperiodic Tasks.

Assume that the application is scheduled using EDF and all aperiodic tasks are scheduled using a Total Bandwidth Server (TBS).

(a) (2 points) What is the maximum utilization of the TBS?

Sample solution:
0.4, as the maximum utilization is 1 and the utilization of periodic tasks is 0.6.

(b) (5 points) Determine the *worst-case* finish time of aperiodic tasks J_1, J_2, J_3.

Sample solution:
Let A_i and F_i denote the arrival and finish time of aperiodic task J_i. In the worst-case, aperiodic tasks finish at their deadlines.

$$\begin{aligned} F_1 &= 57 + 10.5/0.4 &&= 83.25 \\ F_2 &= \max(60, 83.25) + 37.2/0.4 &&= 176.25 \\ F_3 &= \max(180, 176.25) + 65.4/0.4 &&= 343.5 \end{aligned}$$

1.3: Resource Sharing (maximal 13 points)

Three tasks share a single resource protected by a critical section. The execution patterns for the tasks are shown in Figure 3 and their release times are given in Table 3. The critical sections are shaded in grey. P_1, P_2, and P_3 represent the priorities of Task 1, Task 2, and Task 3, respectively. Lower index tasks have higher priority; i.e., $P_1 > P_2 > P_3$.

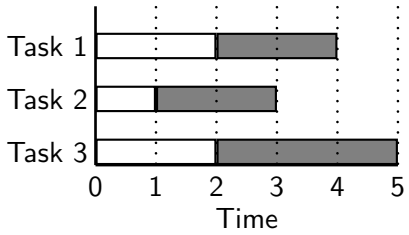


Figure 3: Execution Patterns.

	Release Time
Task 1	5
Task 2	3
Task 3	0

Table 3: Release Times.

(a) (5 points) Use the Priority Inheritance Protocol to schedule all tasks. Draw your schedule in Figure 4.

Sample solution:

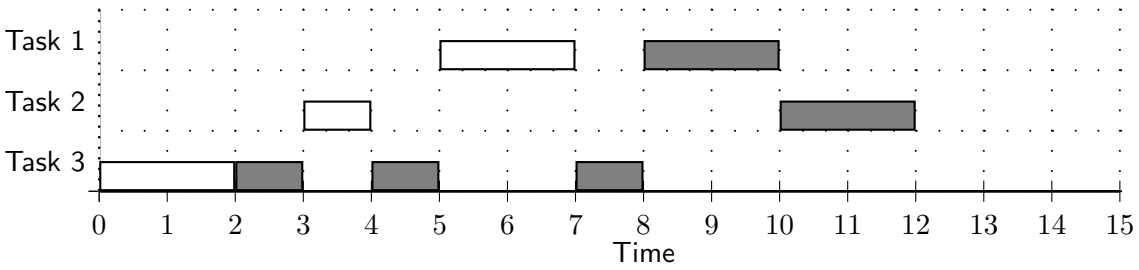


Figure 4: Task Schedule.

(b) (3 points) Draw the priority level of Task 3 with respect to time in Figure 5.

Sample solution:

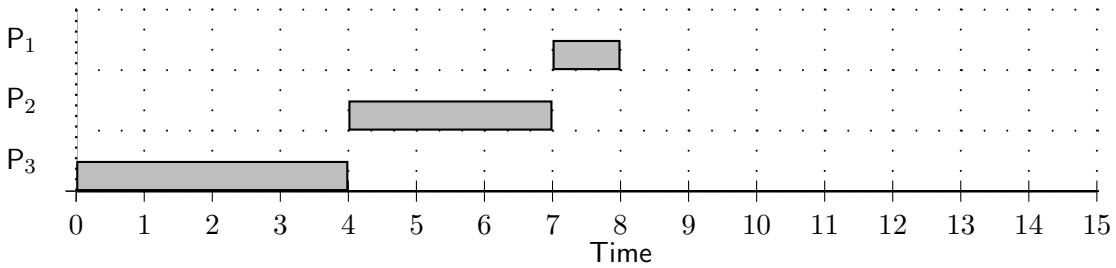
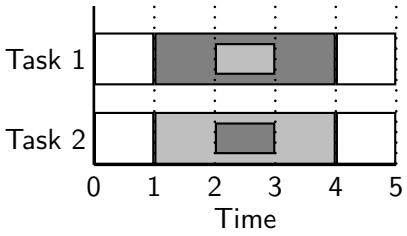


Figure 5: Task 3 Priority.

(c) (5 points) Explain with an example how a deadlock can occur when the Priority Inheritance Protocol is used. (Note: This question is independent of questions 1.3a and 1.3b.)

Sample solution:
This occurs with nested critical sections. Consider following execution patterns:



The release times of Task 1 and Task 2 are 1.5 and 0 respectively. At time 1, Task 2 acquires lock to light gray critical section. Task 2 is preempted at time 1.5 by Task 1, which has higher priority. Task 1 acquires lock to dark grey critical section at time 2.5. At time 3.5, Task 1 tries and fails to acquire lock to light grey critical section; since it is held by Task 2. Task 2's priority increases. It executes from time 3.5 - 4. At time 4, Task 2 tries and fails to acquire dark grey critical section; since it is held by Task 1. Neither task can now make progress.

Task 2 : Components and Communication (maximal 45 points)

2.1: Components (maximal 3 points)

(a) (1 point) Digital Signal Processors are especially suited for ... (check one box).

Sample solution:
Control Dominated Systems ☐ Data Dominated Systems ☒

(b) (1 point) Briefly outline (in about two sentences) the main characteristics of Control Dominated Systems and of Data Dominated Systems.

Sample solution:
Control Dominated Systems are reactive systems which are event driven. Data Dominated Systems are streaming oriented with (mostly) periodic behavior.

(c) (1 point) State in two sentences, what are the main characteristics of FPGAs and ASICs? What are the respective advantages and disadvantages for implementing an embedded application?

Sample solution:
FPGAs are more flexible (reprogrammable); ASICs have a fixed application; ASICs are harder to design, more expensive, but better fitting for specific application (lower energy footprint, increased performance).

□

2.2: CSMA/CR (maximal 6 points)

Consider a wired communication system with Carrier Sense Multiple Access Collision Resolution (CSMA/CR). The system is *dominant high*, meaning that a device sending a high (1) has higher priority than a device sending a low (0). The communication packets consist of [Device ID | Data] (most significant bit sent first); there are four devices A, B, C, and D, with device IDs $A = 12$, $B = 7$, $C = 13$, and $D = 14$.

(a) (4 points) Sort the devices A, B, C, and D in descending order of priority. (Hint: start with the device that has the highest priority in accessing the communication system.)

Sample solution:
Convert the Client IDs to binary representation and compare the binary strings. The solution is: D, C, A, B

□

(b) (2 points) In the system described above, if the highest priority device is continuously sending packets, the other devices are not be able to send any packets. This effect is called *starvation*. Can starvation also happen when using CSMA/CD? Provide a short motivation (about two sentences) for your answer.

Sample solution:
NO. Ethernet also uses timeslots, but in case of an collision, all clients which tried to send wait for an arbitrary amount of time. Due to the random waiting time (back-off), there is no prioritizing and therefore starvation cannot happen.

□

2.3: Token Ring (maximal 18 points)

Consider the token ring with four clients of Figure 6. The maximum transmission rate is $16\,Mbps^{\dagger}$, communication proceeds in rounds, each round can last **at most** $5\,ms$, and there is one token. Within each round, the client that has the token performs the following steps:

1. If there is data ready, send a packet consisting of a $4\,kb^{\ddagger}$ header, followed by data.
2. If a packet was sent, wait until an ACK of $4\,kb^{\ddagger}$ is received.
3. Pass on the $8\,kb^{\ddagger}$ token to the next client.

Step 3 always takes place once per round; steps 1 and 2 only happen, once per round, if the client has data to send. If data is too big to be sent in one round, it has to be partitioned

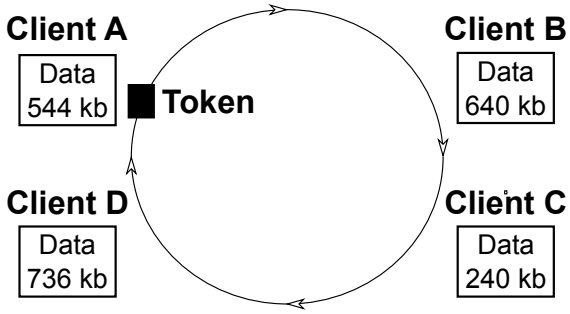


Figure 6: Token ring status at time 0; the four clients need to send the specified data amounts†.

into multiple rounds. All clients receive messages without any delay and the processing time for tokens, packets, and ACKs is negligible. At time 0, client A has the token; at the end of

$^{\dagger}1\,Mbps = 10^6\,bps$ (bits per second), $1\,kbps = 10^3\,bps$
$^{\ddagger}1\,Mb = 10^6\,b$ (bits), $1\,kb = 10^3\,b$

each round, the token is passed to the next client on the network, in a clockwise direction. At time 0, every client has data ready to send; Figure 6 shows the data size.

(a) (12 points) **Calculate** the total time needed until all clients have sent all their packets and the token is in possession of Client A again.

(b) (6 points) **Calculate** the aggregated data throughput of the network (in $Mbps$) for this scenario.

Sample solution:

(a) Overall sending time:

- Sending the token takes $0.5\ ms$, the ACK takes $0.25\ ms$ and the header causes additional $0.25\ ms$ that are needed for the transmission. Therefore the net time that can be used to transmit a packet per round is $4\ ms$. This means a packet of a maximum size of 64 kb can be transmitted.
- Calculate how many rounds are needed per client. The maximum number is 12 rounds, needed by Client D. This means the token has to travel 12 rounds till all the packets are fully sent and the token is back at Client A.
- The token needs $0.5\ ms$ to be sent. Therefore, in order to complete a full round it takes $2\ ms \rightarrow 24\ ms$ for 12 rounds.
- In total 35 packets need to be sent, which causes an additional overhead of $35 * 0.5 = 17.5\ ms$.
- Total Time $= \sum$ (Time needed for packet sending) + Token Time + Packet Overhead $= 176.5\ ms$

(b) $\frac{\sum(\text{Size of Packet})}{\text{Total Time}} = 12.238\ Mbps$

□

2.4: Bluetooth (maximal 18 points)

A Bluetooth connection is defined with the following characteristics:

- Slaves can only send packets directly after they have received a packet from the master.
- One slot has a duration of $625\,\mu s$.

- Each packet contains a $126\,bit$ header, a $24\,bit$ CRC, and the **maximum** integer number of bytes[‡] of payload (user data) to fill up the rest of the packet.
- It is possible to send packets with the length of 1 slot, 3 slots or 5 slots.
- Only a **maximum** of $366\,\mu s$ of the first slot can be used for packet transmission.
- The data rate is $1\ Mbps$[†]

(a) (10 points) Determine the channel throughput (only transmitted user data, excluding header, CRC, etc.) for the packet lengths given in Table 4 and the Bluetooth connection defined above. Please show all your calculations.

Sample solution:

- Packet length in slave direction 1: The packet consists of $366\ bits$ of which $126\ bits$ are reserved for the MAC header and another $3\ bytes$ are used for the higher layers and the CRC. The number of data bits that can be sent in one slot thus is $366-126-3*8 = 216$. Since, master and slave send data in alternate slots, and the length of one slot is $625\mu s$. The throughput in either direction is $\frac{216}{2*0.625} = 172.8\ kbps$.
- Packet length in slave direction 3: DH3 packet structure allows sending of $183\ bytes$ ($1464\ bits$), and spreads across 3 $625\mu s$ slots. After every three $625\mu s$ slots used by the slave, the master uses one $625\mu s$ for polling. Thus, the throughput in the slave direction is $\frac{1464}{4*0.625} = 585.6\ kbps$. The master continues to send $216\ bits\ per\ slot$, every 4^{th} slot with a throughput of $\frac{216}{4*0.625} = 86.4\ kbps$.
- Packet length in slave direction 5: DH5 packet type allows sending $339\ bytes$ ($2712\ bits$) in five $625\mu s$ slots. Thus, the number of data bits sent is $\frac{2712}{6*0.625} = 723.2$. The argument for the throughput in the master direction is same as above, which leads to a throughput of $\frac{216}{6*0.625} = 57.6\ kbps$.

Packet Length in Slots		Throughput in kbps (1 kbps = 1000 bps)	
To Slave	To Master	To Slave	To Master
1	1	172.8	172.8
1	3	86.4	585.6
1	5	57.6	723.2

Table 4: Packet lengths and channel throughput calculation.

[‡] $1\,byte = 8\,bits$
[†] $1\ Mbps = 10^6\ bits\,per\,second$

(b) (2 points) Why is the first $625\,\mu s$ slot not entirely used for packet transmission? Answer in one short sentence.

Sample solution:
The spare time needs to be used to prepare the hardware for the next slot because the frequency is changed.

(c) (6 points) Which two connection types are defined in the Bluetooth Standard, and what are those types used for? Please state in two short sentences.

Sample solution:

- Synchronous Connection-Oriented - Point-to-Point Full Duplex in reserved slots - regular data transmission (big data)
- Asynchronous Connection-Less - Asynchronous Service without reserved slots - spontaneous transmissions/irregular

Task 3 : Low Power Design (maximal 35 points)

3.1: Energy Minimization (maximal 18 points)

Consider a heterogeneous dual-core platform consisting of processors π_1 and π_2. A task of 10^7 clock cycles is going to be executed on this platform. The workload can be freely partitioned between the two processors to run in parallel. In addition, we make the following assumptions:

- The platform has two operation modes: *active* and *sleep*.
- In active mode, the frequency, dynamic and leakage power consumptions of π_i are denoted as f_i, P_{di} and P_{li}, respectively. We assume they are constant: $f_1 = 1$ MHz, $P_{d1} = 3$ mW, $P_{l1} = 0$ mW, $f_2 = 4$ MHz, $P_{d2} = 10$ mW and $P_{l2} = 6$ mW. We assume dynamic power is only dissipated when a processor is processing tasks, while leakage power is always dissipated in active mode.
- In sleep mode, the processor frequency is zero and no power is dissipated, neither dynamic nor leakage. The switching overheads in terms of time and energy can be neglected.

(a) (10 Points) Assume that a processor can switch to sleep mode whenever it is idle, i.e., when it is not processing tasks. What is the optimal (i.e., minimal) energy consumption of the system?

Sample solution:
We can first derive the energy consumption when executing a single clock cycle:

$$E_{singlecycle,i} = t_{singlecycle,i} \cdot P_{overall,i} = \frac{1}{f_i} * (P_{di} + P_{li})$$

- For π_1: $E_{singlecycle,1} = \frac{(P_{d1}+P_{l1})}{f1} = 3\,\text{nJ}$. - For π_2: $E_{singlecycle,2} = \frac{(P_{d2}+P_{l2})}{f2} = 4\,\text{nJ}$.

Processor π_1 is more energy-efficient. Hence, the solution is to let the task run completely on π_1 for $\frac{10^7}{10^6 Hz} = 10\ s$. Then put π_1 into sleep mode. π_2 is not utilized and resides in sleep mode. Total consumed energy is $10\,\text{s} \times (P_{d1} + P_{l1}) = 30\,\text{mJ}$

□

(b) (8 Points) Assume that the platform can only switch to sleep mode if *both* processors are idle. What is the optimal (i.e., minimal) energy consumption of the system now? (**Hint**: assume that x and y number of clock cycles are assigned to π_1 and π_2, respectively. You can formulate and solve the problem with respect to x and y, or find directly the relation between x and y.)

Sample solution:

$$\begin{aligned} \min \quad & \frac{x}{1\text{MHz}} \times 3 \text{ mW} + \frac{y}{4\text{MHz}} \times 10 \text{ mW} + \max\{\frac{x}{1\text{MHz}}, \frac{y}{4\text{MHz}}\} \times 6\text{mW} \\ s.t. \quad & x + y = 10^7,\ \{x, y\} \in \mathbb{N}_0^2 \end{aligned}$$

Now we can distinguish between two cases $x \geq \frac{y}{4}$ or $x < \frac{y}{4}$. We find the optimal solution exists when $x = 2 \times 10^6$ and $y = 8 \times 10^6$ (both are active for the same amount of time), and the min. energy is 38 mJ.

□

3.2: Energy Harversting (maximal 17 points)

Consider a processor with negligible leakage power dissipation and dynamic power dissipation specified as $\mathsf{P}_{\mathsf{dynamic}} = \left(\frac{\mathsf{f}}{\mathsf{MHz}}\right)^3$ mW, where f is the frequency in Hz. The processor is put into a zero power state whenever it is idle. The set of hard real-time tasks of Table 5 needs to be executed on the processor:

Tasks	T_1	T_2	T_3
Period (ms)	6	4	12
Relative Deadline (ms)	6	4	12
Cycles (x10^3)	2	1	2

Table 5: Characteristics of the hard real-time tasks to be executed.

All tasks initially arrive at time zero. The system has a battery with initial energy C microjoule (μJ) and it is replenished by a constant power source of A microjoule per millisecond (μJ/ms).

(a) (9 Points) Assume $C = 6$ μJ, $A = 0.5$ $\frac{\mu\mathsf{J}}{\mathsf{ms}}$ and $f = 1$ MHz. Apply EDF scheduling and draw in Figure 7 the battery energy during the time interval $[0$ ms$, 12$ ms$]$.

Sample solution:
Execution times of T_1, T_2 and T_3 are 2 ms, 1 ms and 2 ms, respectively. Applying EDF scheduling, the processor is busy in $[0$ ms$, 9$ ms$]$.

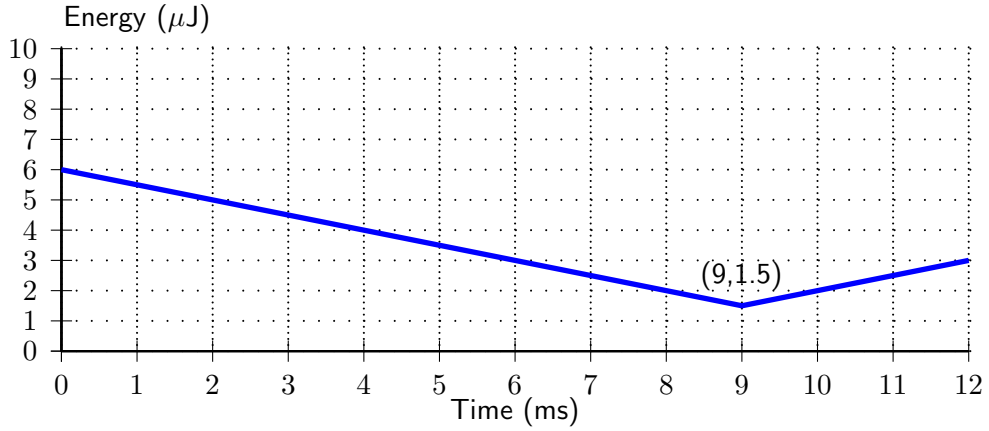


Figure 7: Battery energy diagram.

□

(b) (8 Points) Assume the battery does not run out of charge. Prove or disprove the following statement:

> To have the maximum possible battery energy after each hyper-period (12 ms), all tasks should run at the same frequency.

(**Hint:** providing main arguments or formal proof are both accepted.)

Sample solution:
Method 1:
Applying the statement in the slides, (1) run on a constant frequency, (2) fully utilize the processor, (3) above two minimizes energy consumption for each hyper-period, (4) remaining energy is maximized.

Method 2:
For (1) and (2), applying YDS will also lead to the same conclusion.

□

Task 4 : Architecture Synthesis (maximal 60 points)

4.1: Architecture Synthesis Fundamentals (maximal 12 points)

Mark the following statements as *true* or *false* and provide a brief explanation (1 sentence).
Sample solution:

- (1 point) The different paths in a dependence graph represent branches in the control flow of a program.
  ☐ True ☒ False
  Explanation: They represent a partial order among operations of the program. They expose the degree of parallelism of the program operations.

- (1 point) In a marked graph, a node with more than one input edge is activated (it can fire) if there is a token on at least one of its input edges.
  ☐ True ☒ False
  Explanation: It is activated if there is at least one token on **all** input edges.

- (1 point) As-Soon-As-Possible (ASAP) is an *exact* scheduling algorithm for minimizing the latency of an operation set, under *no* resource constraints.
  ☒ True ☐ False
  Explanation: It is no heuristic approach. It guarantees minimal latency when there are no resource constraints.

- (1 point) Under resource constraints, the LIST algorithm returns a schedule with guaranteed minimal latency.
  ☐ True ☒ False
  Explanation: LIST is a heuristic approach. ILP would return an optimal solution.

- (2 points) The LIST scheduling algorithm can be applied to pipelined implementations with limited resources.
  ☒ True ☐ False
  Explanation: It can be modified to account for resources that are still occupied by operations from previous iterations.

- (2 points) When loop folding is enabled in a functional pipeline, operations are scheduled such that their starting and finishing times are always in the *same* physical iteration.
  ☐ True ☒ False
  Explanation: With loop folding, the starting and finishing time of an operation can be in different physical iterations.

- (2 points) Consider the sequence graph of Fig.8(a) and the resource graphs of Fig.8(b). A unit of each resource type (ADD or MUL) costs 0.5 SFr. Given these, there is exactly *one* resource allocation that minimizes *both* the schedule latency and implementation cost.

  ☒ True ☐ False

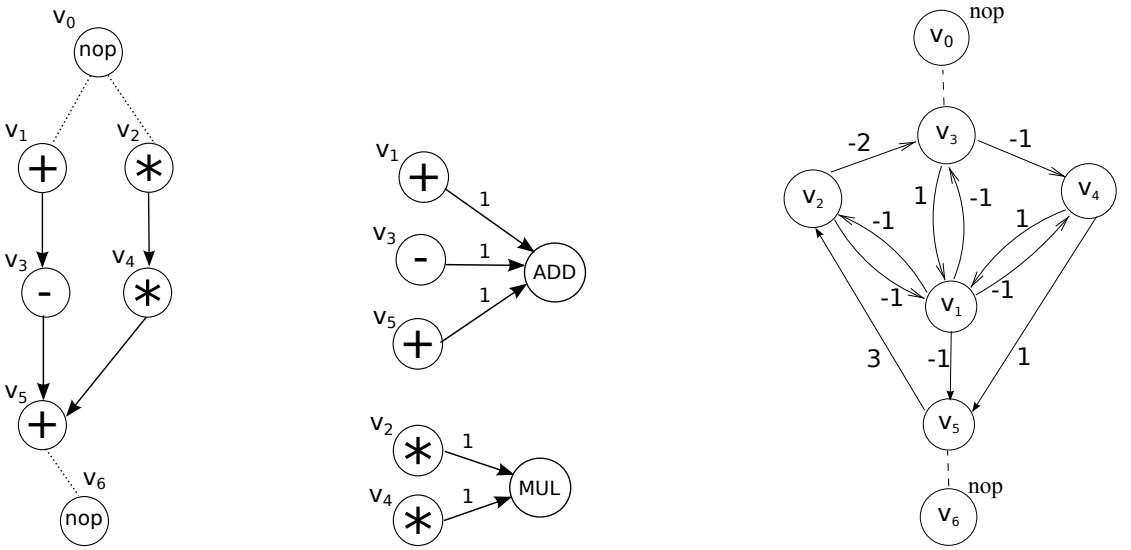
  Explanation: In this case, the Pareto-front consists of only 1 solution (latency=3, cost=1).

- (2 points) For the weighted constraint graph of Fig.8(c), there is a feasible schedule which fulfills all timing constraints.

  ☐ True ☒ False

  Explanation: V1 $\rightarrow$ V5 $\rightarrow$ V2 : $\sum = +1$ (positive cycle).

☐



(a) Sequence Graph. (b) Resource Graph. (c) Weighted Constraint Graph.

Figure 8: Architecture Synthesis Fundamentals

4.2: Scheduling with Resource Constraints (maximal 25 points)

Determine a resource allocation and a schedule for implementing the following functions:

```
int func (int a, int b, int c, int d, int e, int f, int g, int h){
      int x = (a *1 b) +3 (c *2 d) +4 2;
      int y1 = (e +5 f);
      int y2 = (g +6 h);
      int y = mod (y1, y2);
      int z = (x *8 y);
      return z;
}

int mod (int arg1, int arg2){
      int res = arg1 %7 arg2;
      return res;
}
```

Notation $*_1$ indicates that the index of this multiplication is 1 (node v_1 in a sequence graph).

Additions ($+$) need **1 cycle** to be executed on an adder, ADD. Multiplications ($*$) and modulo operations ($\%$) need **2 cycles** on a multiplier, MUL. Calling a function and returning take **0 time**.

(a) (6 points) Provide the hierarchical sequence graph $G_S = (V_S, E_S)$ for function **func**. Denote each node as v_i, where i is the index of the corresponding operation in the code.

Sample solution:
See Fig.9.

□

(b) (6 points) Apply the ASAP and ALAP algorithms to compute the earliest (l_i) and latest (h_i) starting times of all operations $v_i \in V_S$, $i \in \{1, \cdots, 8\}$. For ALAP, assume the maximum latency $\bar{L} = 7$. Fill in the starting times in Table 6.

Sample solution:
See Table 6.

□

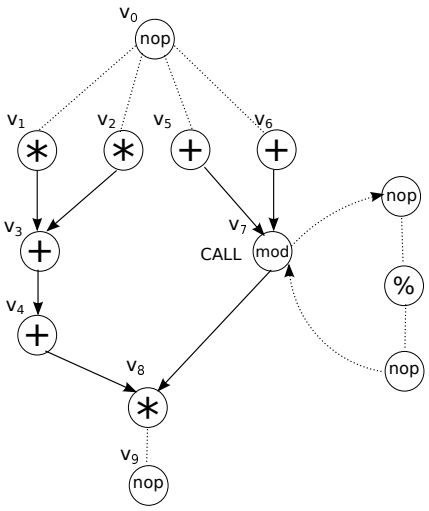


Figure 9: Sequence graph G_S

	$l_i(ASAP)$	$h_i(ALAP)$
v_1	0	1
v_2	0	1
v_3	2	3
v_4	3	4
v_5	0	2
v_6	0	2
v_7	1	3
v_8	4	5

Table 6: Starting time bounds given no resource constraints

(c) (13 points) An Integer Linear Program (ILP) needs to be formulated for the problem of *area minimization* of the implementation, under latency constraints. On a given chip, an adder (ADD) requires $s(\text{ADD}) = 0.15\text{mm}^2$ and a multiplier $s(\text{MUL}) = 0.35\text{mm}^2$, respectively. We seek a resource allocation and a feasible schedule with latency not greater than $L_{max} = 8$.

	v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8
l_i	0	0	2	4	0	0	2	6
h_i	2	2	4	5	2	2	4	6

Table 7: Starting time bounds for operations v_1, ..., v_8

For the ILP formulation, the binary variables $x_{i,t} \in \{0,1\}$ are defined $\forall v_i \in V_S$ and $\forall t : l_i \leq t \leq h_i$. Please consider the l_i, h_i values in Table 7 for this question (**not** the values from (b)). $x_{i,t} = 1$ if operation v_i starts executing at time t in a schedule or $x_{i,t} = 0$, otherwise. Function $\tau : V_S \rightarrow \mathbb{N}$ can be used to denote the starting time of an operation $v_i \in V_S$ and function $\alpha : V_R \rightarrow \mathbb{N}^+$ to denote the allocation of resource instances, where $V_R = \{\text{ADD}, \text{MUL}\}$. Given the above notations:

- (4 points) Express the objective function of the ILP.

  Sample solution:
  minimize: $s(\text{ADD}) \cdot \alpha(\text{ADD}) + s(\text{MUL}) \cdot \alpha(\text{MUL})$

  □

- (2 points) Express the latency constraint.

  Sample solution:
  $\tau(v_9) - \tau(v_0) \leq 8$

  □

- (2 points) Define $\tau(v_1)$ as a function of $x_{1,t}$, where $l_1 \leq t \leq h_1$.

  Sample solution:
  $\tau(v_1) = x_{1,1} + 2 \cdot x_{1,2}$

  □

- (5 points) Express all resource constraints at time $t = 2$ (without $\sum$ formulation).

  Sample solution:
  $t = 2$:

$$\begin{aligned} x_{1,1} + x_{1,2} + x_{2,1} + x_{2,2} + x_{7,2} \leq &\quad \alpha(\text{MUL}) \\ x_{3,2} + x_{5,2} + x_{6,2} \leq &\quad \alpha(\text{ADD}) \end{aligned}$$

□

4.3: Iterative Algorithms (maximal 23 points)

Consider the marked graph G_M in Figure 10. The nodes labelled with **+**, *****, **x2** represent addition, multiplication of two input values, and multiplication of an input value by 2, respectively. The edge $f_3 \to f_2$ has m initial tokens, and the edge $f_3 \to f_1$ has n initial tokens, where $m > n$. The input u generates a sequence of numbers, with $u(k)$ being the k-th number.

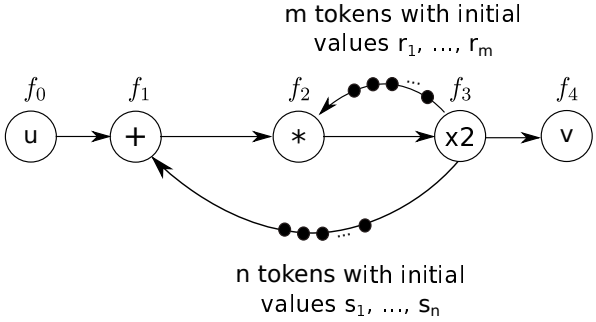


Figure 10: Marked graph G_M

(a) (6 points) Determine the output value $v(k)$ as a function of the input values for $k > m$.

Sample solution:
$v(k) = 2 \cdot (u(k) + v(k-n) \cdot v(k-m)$

□

(b) (6 points) Assume $m = 3$ and $n = 1$. Illustrate the data dependencies among the operations with an equivalent extended sequence graph $G_S = (V_S, E_S, d)$, where $V_S = \{f_0, \cdots, f_4\}$ and d_{ij} denotes the index displacement for each $(f_i, f_j) \in E_S$.

Sample solution:
See Figure 11.

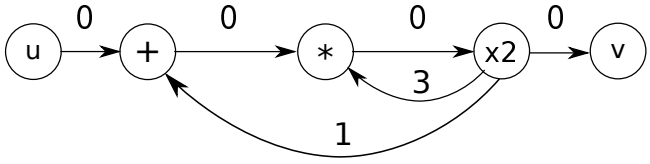


Figure 11: Extended sequence graph G_S

□

(c) (6 points) An addition needs **1** time unit and a multiplication needs **2** time units to execute. Suppose that we implement the algorithm using functional pipelining. Express all data dependency constraints in G_S in the form:

$$\tau(f_j) - \tau(f_i) \geq w(f_i) - d_{ij} \cdot P, \quad \forall (f_i, f_j) \in E_S, \tag{1}$$

where P is the iteration interval of the pipeline and $w(f_i)$ the execution time of f_i.

Sample solution:
Data dependency constraints:

$$\begin{array}{rll}
\tau(f_1) \geq & 0 & \\
\tau(f_2) - \tau(f_1) \geq & 1 & (1) \\
\tau(f_3) - \tau(f_2) \geq & 2 & (2) \\
\tau(f_4) - \tau(f_3) \geq & 2 & (3) \\
\tau(f_1) - \tau(f_3) \geq & 2 - P & (4) \\
\tau(f_2) - \tau(f_3) \geq & 2 - 3 \cdot P & (5)
\end{array}$$

□

(d) (5 points) Assuming unlimited resources, what is the highest throughput $\frac{1}{P}$ that can be achieved with functional pipelining?

Sample solution:
Based on the previous system of inequalities:
(1)+(2)+(4) $\implies 0 \geq 5 - P \implies P \geq 5$
(2)+(5) $\implies 0 \geq 4 - 3P \implies P \geq \frac{4}{3}$
Hence, $P_{min} = 5$ and the max. feasible throughput is 1/5.

□