Prof. Bernd Finkbeiner, Ph.D.
Dipl.-Inf. Rüdiger Ehlers
Markus Rabe, M.Sc.
Sebastian Hahn, B.Sc.

Summer term 2012
Problem Set 10
Due: Wednesday, 18th July 2012

Embedded Systems

Please indicate your **name**, **matriculation number**, **email address**, and which **discussion session** you have been allocated to. We encourage you to collaborate in **groups** of up to **three** students. Only one submission per group is necessary.

Exercise 1: Periodic multiprocessor scheduling

Consider the following periodic task set:

Task i	**Computation time** C_i	**Period** T_i
1	6	8
2	7	8
3	6	8
4	5	8
5	8	8

Assume a four-processor architecture with a task migration cost of **one** time unit. Find a feasible *periodic* schedule.

Exercise 2: Hardware/Software Partitioning

A set of function objects (or tasks) $O = \{o_1, \ldots, o_n\}$, $n \in \mathbb{N}$, can either be implemented in hardware (HW) or software (SW). Each object o_i, $1 \leq i \leq n$, has HW costs $c_h(i)$, SW costs $c_s(i)$, HW computation time $d_h(i)$, and SW computation time $d_s(i)$.

(a) Encode the described partitioning problem as an integer programming problem, where cost and computation time are weighted "u USD per second" in the cost function.

(b) Extend your problem such that the total number of function object implementations in HW must not exceed H_{max}.

(c) Extend your problem such that the total costs must not exceed C_{max} and the total computation time must not exceed D_{max}.

(d) Extend your problem such that some pairs of objects must be implemented either both in HW or both in SW. Assume that these pairs are given by a relation $\{(o_1, o'_1), \ldots, (o_m, o'_m)\} \subseteq O \times O$, $m \in \mathbb{N}$.

Exercise 3: Static Redundancy

Figure 1 shows the *triple modular redundancy* arrangement. It uses three identical parallel working modules, and works correctly as long as at least two of the three modules are intact.

1. Assume that all three modules have the same reliability $R(t)$. Also assume that the voter is absolutely reliable. Prove that under these assumptions the reliability of the TMR arrangement is
$$R_{TMR}(t) = 3R^2(t) - 2R^3(t).$$

2. Is the realiability of the TMR arrangement always higher than that of a single module? Justify your answer!

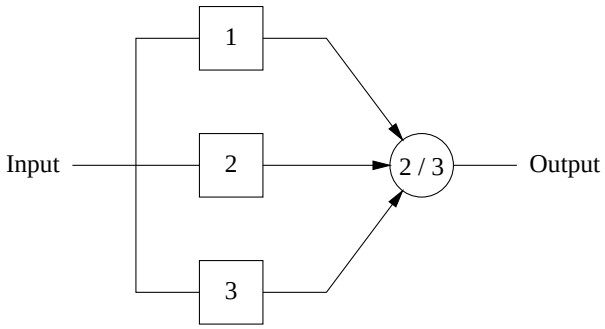
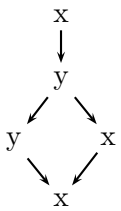


Figure 1: TMR arrangement.

3. When a module has a constant failure rate λ then its reliability falls exponentially according to the law $R(t) = e^{-\lambda t}$. At what point of time is the deployment of the TMR versus one single module no more justified?

Exercise 4: Cache analysis and predictability

We consider two cache replacement strategies: *least recently used* (LRU) and *first in first out* (FIFO). For both strategies, the 'oldest' cache entry is replaced, but for LRU and FIFO, the 'age' is determined in different ways. In each step the age of all entries is increases by one and in case of a cache miss, the new entry comes in with age 0. The strategies differ only in case of cache hits: For LRU the age of the cache entry that was used in this step is set to 0, while for FIFO the age is not resetted. Consider the following sequence of accesses:



A split of the branches, as in the graph above, indicates that both options could happen and that static analysis of the code cannot rule out any of the two options.

1. Assume an LRU-cache with 2 entries. Apply a must-cache and a may-cache analysis of the program. Can you predict the exact cache state at the last access to x?

2. Assume a FIFO-cache with 2 entries and an empty cache at the beginning. Can you predict a cache-hit or a cache-miss at the last access to x?

3. Repeat (2) with an unknown cache-state at the beginning of the program. What can you say about the predictability of an LRU-cache compared to a FIFO-cache?