EE249 Homework 5

TA: Alessandro Pinto
{apinto@eecs.berkeley.edu}

November 11, 2003

Goal of HW5 is to expose you to some mapping related problems. General Comments: there is not right/wrong answer to a question, I would rather consider your homework as solutions to design problems. I could argue that your solution is inefficient and you could argue the same about mine. Grade is established mostly on the rationale you follow to answer questions. Hence it is in your interest to justify all your statements. You can use any kind of sources as long as you reference them all.

Homework are created by the TA from scratch. It is then reasonable that he may forgot assumptions or make some that are not necessary. You can add or remove assumptions at you convenience as long as you justify your decisions.

Problem description: In this homework I will let you a lot of freedom in terms of design choices. Consider again the elevator problem. Figure 1 shows a block diagram of the entire system.

Keypads at different floors (W), and in the elevator (R), are connected to a centralized user interface $UI$. This unit has the specific task of collecting and storing all the requests and sending them to a $Strategy$ function that has to decide the next action to take. In particular the strategy function will decide the next target floor $f_t$. This information is sent to the $TransitionControl$ block that has a big role in the entire system. It is basically a finite state machine that acts informally as follows:

- send a close signal to the $DoorControl$ block; if the isclosed signal is received the proceed to the following step, while if a reopen signal is received then send an open signal, wait until an isopen signal is received and try to close the door again.

- Send the target height $x_t$ to the $ElevatorControl$ block.

Let’s spend few word on the ElevatorControl. It implements the state machine that you have described in HW4. In each state it asks to the four controllers in the left of figure 1 to generate a new control sample for the elevator motor. So in each state there is a periodic activation of ElevatorControl sending a control signal to the controller which generates a new controlled variable
sample and answers with a done signal. When the ElevatorControl changes state, going for instance from Acceleration to Cruise, it also changes the control algorithm by sending control signal to a different block.

Passengers in the elevator can use the (R) keypad to ask for different actions. All of them are the same as for the (W) keypad, but one. There is an halt button that a passenger can use to ask the elevator to stop. This is like an emergency button and we want the elevator to react right away within a bounded latency \( l_e \).

Figure 2 shows the block diagram of the architecture we want to use to implement our elevator control. You will have to decide an optimal mapping of the function in figure 1 onto the architecture in figure 2. Optimal mapping is with respect to a cost function that I will define later.

The architecture has two processors and one memory. They all communicate through a bus. Each CPU has a real time operating system on top of it that implements our concurrency. You can choose the communication between tasks mapped on the same processor and task on different processor, and you may assume all the communication to be use a shared memory scheme for simplicity.

The user inputs are handled depending on their nature: the (W) keypads are polled by the UI while the (R) keypad interacts through interrupts.

In this example we are pretty lucky because somebody has done the estimation of the execution time that are reported in table 1.

By looking at the table you can find a bound on the cpu clock speed depending on the mapping.
Figure 2: Block diagram of the architecture

<table>
<thead>
<tr>
<th>Task</th>
<th>Period(sec)</th>
<th>CPU1(cycles)</th>
<th>CPU2(cycles)</th>
</tr>
</thead>
<tbody>
<tr>
<td>UI</td>
<td>0.3</td>
<td>50</td>
<td>50</td>
</tr>
<tr>
<td>Strategy</td>
<td>0.3</td>
<td>200</td>
<td>240</td>
</tr>
<tr>
<td>TransitionControl</td>
<td>0.03</td>
<td>100</td>
<td>130</td>
</tr>
<tr>
<td>ElevatorControl</td>
<td>0.01</td>
<td>160</td>
<td>180</td>
</tr>
<tr>
<td>Acc</td>
<td>(*)</td>
<td>500</td>
<td>550</td>
</tr>
<tr>
<td>Cruise</td>
<td>(*)</td>
<td>300</td>
<td>340</td>
</tr>
<tr>
<td>Dec</td>
<td>(*)</td>
<td>500</td>
<td>550</td>
</tr>
<tr>
<td>Landing</td>
<td>(*)</td>
<td>300</td>
<td>340</td>
</tr>
<tr>
<td>DoorControl</td>
<td>(*)</td>
<td>280</td>
<td>300</td>
</tr>
<tr>
<td>HaltISR</td>
<td>(*)</td>
<td>20</td>
<td>20</td>
</tr>
</tbody>
</table>

Table 1: Tasks execution time. (*) means that the block is not periodic but reactive. HaltISR is the interrupt service routine relative to the halt button.

**QUESTION 1:** (Might be difficult) Based on the data you have, find a mapping that minimizes both the clock speed and the latency $l_e$. In order to achieve this you have to decide few things (not necessarily in this order):

- Blocks partitioning between the two CPU’s
- Scheduling policy of the two task set
- Task priorities

In this question you neglect the communication overhead and the RTOS overhead.

Now we will consider the overhead introduced by the RTOS and the communication system. The data are reported in table 2.
Let’s by now consider the RTOS overhead only.

**QUESTION 2:** Re-evaluate your mapping introducing the RTOS overhead. Is still your mapping within the specification? If not, how would you change it?

**QUESTION 3** Making assumptions on the data types, estimate the data bus traffic assuming a Harvard architecture.

**Acknowledgment** I would like to acknowledge Claudio Pinello for helping me in preparing this homework.