Towards Data Path Analysis Using Uppaal *

Georgeta Igna

Radboud University Nijmegen, The Netherlands
g.igna@cs.ru.nl

Abstract. In this paper, we describe the results of modeling and analyzing the first use case provided by Océ, the industrial partner in the Octopus project. The aim of the project is to have a faster way of detecting the impact of changes into the architecture of a multifunctional printer. Our approach was to use timed automata. At this stage, we needed to create a model which was stable enough with respect to the specifications and re-usability. Also, we needed to analyze the scheduling problems which appear in practice for this use case.

1 Introduction

Octopus is a joint research project of the Dutch Embedded System Institute, three Dutch universities and Océ, an industrial partner. Océ, is a producer of digital document printers. The objective of the project is to make a usable model to predict resource usage of the data path for multiple scenarios, so that a designer/architect can evaluate and compare several architectures for these scenarios. In order to make printers more adaptive it is vital that the data path also becomes much more flexible. Data path in this case is the whole trajectory of digital images from the data source, that is, the scanner camera to the print unit where the toner image is formed. In a multifunctional device, it is possible that the data path is used by different scenarios at the same time. It is hard to find a good architecture and to dimension the resources so that a good compromise is achieved between performance in all cases and cost-price. The academic partners utilise three different techniques: timed automata, colored petri nets, and synchronous data flow. In this paper we are going to present the timed automata approach. For dealing with project challenges, we used Uppaal [1,2], a timed automata model checker which fits very well for situations where time is an important constraint.

Because the project is at the beginning, we have tried to follow other solutions with similar problems. A similar case study was carried out by Martijn Hendriks et al. [3] in their work on model checking a controller for a wafer scanner. Also, Ansgar Fehnker's thesis "Guiding and Cost-Optimality in model checking of timed and hybrid systems"[4] is very helpful for those who want to apply model checking techniques for industrial problems.

* Research carried out in the context of the Octopus project, with partial support of the Netherlands Ministry of Economic Affairs under the Senter TS program.

The paper is structured as follows. In the next section we will describe the specifications we had for the this increment of the project. In the third section we explain the modeling decisions we have taken and the solution adopted in order to analyse the timed automata model. The last section contains the conclusions and some directions for future research.

2 Use Case Description

The first use case we had to challenge in the Octopus project is called 'Direct Copy'(Figure 1). As the name suggests, this use case is used whenever a user wants to scan a file and print it. We can observe in the figure that everytime a file is sent to the printer, it is also saved on a disk. This has the role of shortening the time required for reprinting actions.

The arrows show the order in which the components process the image. As we can observe, memory is used for storing the output of some devices. There are some strict rules for avoiding situations when the next device in line reads inconsistent data from the memory. One of the rules refers to the communication between SCAN IP and IP1. IP1 has higher throughput than SCAN IP, thus there must be a delay before it is started. About the moment when IP2 may be started, it is specified that IP2 can start working on a bitmap only after IP1 has completed processing that bitmap. The last rule refers to the last part of the architecture. PRINT IP / USB Client can start reading bitmap while IP2 is writing that bitmap because IP2 has higher throughput than both PRINT IP and USB Client. In adition, we have to ensure that each device has a time limit to process the job.

We are supposed to find a way of detecting the shortest latency between successive printing actions. The way we investigate this issue will be described in the next section.

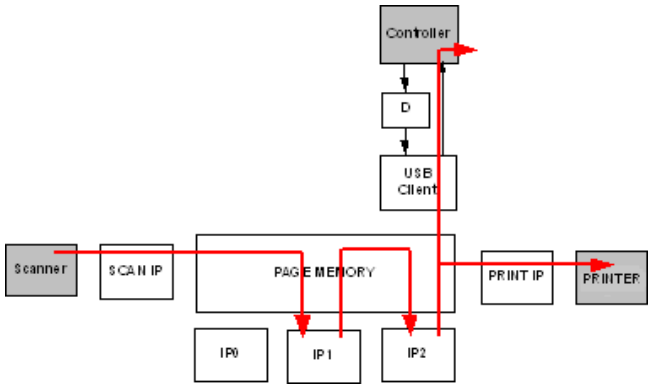

Fig. 1. 'Direct Copy' Use Case

3 Modeling and verification

The automata used for modeling the use case can be classified into two categories. The first category includes the automata designed for the devices from the data path, except for the memory. They have two locations: *run* and *idle* (Figure 2). Whenever a use case starts to claim a resource, it enables the transition between INIT and RUN. In the mean time, this client sets the value for the execution time of the resource(ip1_time in figure 2). The memory is modeled, for the moment, using a global variable. Also, usb bus is a component not modeled yet, but in the next increments, we are going to give it a special attention. As we mentioned in the introduction, both memory and usb bus are the source for the bottlenecks in the system.

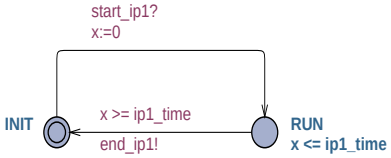

Fig. 2. IP1 automaton

In the second category, there are included the automata used for modeling the use case particularities. One automaton from this category is the Use Case automaton. The others are the two automata used as an interface between the Use Case automaton and the two peripherals: PRINT IP and Controller. The Use Case automaton can be seen in figure 3. Parameters for the use case are: the settings for the input and the output media size(A4 or A3) and, the media type(colored or black and white). Use Case automaton has the following actions: allocates the memory required by the devices from the path, starts(using channels) each automaton from data path, computes and sets(using functions) the execution time for each device automaton, and whenever these devices finish their execution time it decreases the memory.

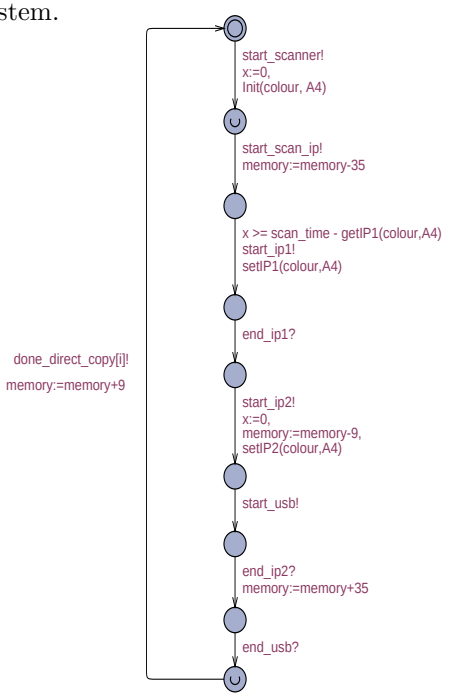

Fig. 3. Use Case automaton

Using the interface automata, we simulate parallel execution of resources from the second part of the use case data path. The printer and usb bus cannot always be used in the same time. The use case automaton sends a request via channels to the interface automaton(Figure 4) that it needs to use a resource. It is the interface automaton job to wait before the target resource is not used and to send the use case request to this resource. Also, this third party automata monitors the moment when the shared resource

has finished its execution time(i.e. in Figure 2, the transition between RUN and INIT is enabled), and sends signals to use case automaton about this(i.e. in Figure 4, join_print! synchronisation is used).

As we mentioned in Section 2, we intend to analyze the delay boundaries between consecutive printing processes. Therefore, we will use a common trick for Uppaal users, a third party automaton(*Observer automaton*). This automaton(Figure 5) has two transitions: one between Init and Final locations, which is fired when the first paper is printed, and the Final self transition, which is fired for next successive printing actions. Every time a document is printed, an internal clock x is reset. This clock is used to get the time needed for completing a job.

For concurrent situations, we have multiple instances for the use case automaton, each instance with its own observer and own interface automaton. Using the query language we can interrogate Uppaal to find out the moments when the first paper comes out and the next printing events. The query is the following:

$$\text{E[] (O.INIT imply O.x} \leq L1) \wedge (\text{O.FINAL imply O.x} \leq L2), (1)$$

where O is the observer automaton for one use case instance, L1 and L2 are the searched limits, x represents an internal clock from observer automaton. After some iterative interrogations we found out that these limits depend only on the scanner and printer devices performances.

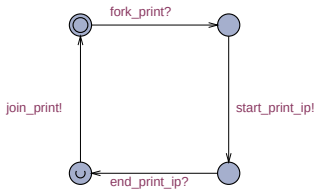

Fig. 4. 'Use case print' automaton

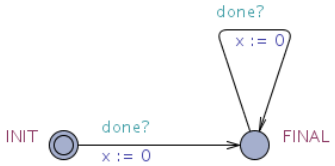

Fig. 5. Observer automaton

Considering two A4 and A3 jobs created periodically for the direct copy use case, the shortest latency is depicted in Figure 6. A4_Rep and A3_Rep are used for making a difference between the first printing operation and the rest of printing actions. In this chart we can observe that IP1 is called after scanning process is finished. This means that the execution time for this device does not influence the final result. Also, in the second and the third PRINT_IP blocks, A4 jobs are printed immediately after the A3 jobs are finished.

4 Conclusions and Further research

In this increment, using Uppaal we could model and detect the shortest latency for the 'Direct Copy' use case. The experiments showed that the limits for both first time printing and successive printing depend on the scanner and the printer performances.

Major challenges in our research are to handle multiple concurrent use cases, and support for time decision whether a new use case may start. Usb and memory buses are the next parts which need to be modeled. Furthermore, we expect that memory operations need to be placed in a separate automaton and modeled in a different way if we want to make a more refined model of memory management including issues like fragmentation.

At this stage, Uppaal has been a good approach for analyzing the system, but in the future when we plan to have many concurrent use cases, state space explosion may appear. And therefore it may be the case that we need to use other approaches in our project in order to eliminate this problem.

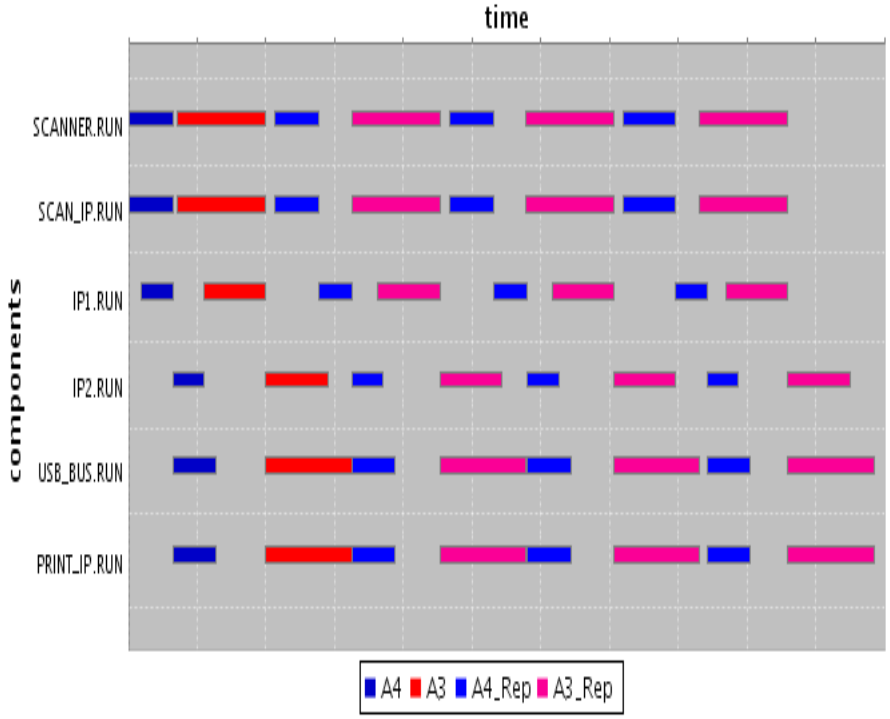

Fig. 6. Succesive printing: an A4 job followed by an A3 job

References

[1] Larsen, K. G., Pettersson, P., Wang, Y.: Uppaal in a nutshell. International Journal on Software Tools for Technology Transfer (1997) 134–152.

[2] Uppaal homepage: www.uppaal.com

[3] Hendriks, M., Nieuwelaar van den, N.J.M., Vaandrager, F.W., Model Checker Aided Design of a Controller for a Wafer Scanner,International Journal on Software Tools for Technology Transfer (STTT). Special Section on Quantitative Analysis of Real-time Embedded Systems (2006) 633–647

[4] Fehnker, A.: Guiding and Cost-Optimality in model checking of Timed and hybrid systems. Ph.D. Thesis (2002)