TERMINAL QOS MANAGEMENT ON RUN-TIME RECONFIGURABLE PLATFORMS

N. Pham Ngoc[1], G. Lafruit[2], G. Deconinck[1], and R. Lauwereins[2]

[1]Katholieke Universiteit Leuven-ESAT/ELECTA
Kasteelpark Arenberg 10
B-3001 Leuven-Heverlee, Belgium
{Nam.Phamngoc, geert.deconinck}@esat.kuleuven.ac.be
[2] IMEC-DESICS, Kapeldreef 75, B-3001 Leuven-Heverlee, Belgium
{lafruit, lauwerei}@imec.be

ABSTRACT

The paper presents a Terminal QoS middleware for run-time reconfigurable platforms which consist of an instruction set processor and a run-time reconfigurable hardware. The main function of the QoS middleware is to decide which tasks should be put in SW, which tasks should be put in HW and at which quality level in such a way that the user receives the highest possible quality of the application given the resource constraints. This decision-taking problem is formulated in terms of a NP-hard optimisation problem for which an approximate solution using a genetic algorithm is proposed.

1. INTRODUCTION

Advanced multimedia applications such as Mpeg-4 applications typically share common characteristics, viz. the need to be able to access a wide variety of multimedia content using a heterogeneous communication and consumption infrastructure, in combination with low cost and low power requirements.

The fact that a large variety of heterogeneous multimedia content has potentially to be dealt with (depending on user preferences and interaction) can lead to a cost inefficient over-dimensioning of network and terminal resources. To tackle this issue, advanced resource management techniques are needed that make trade-offs on the fly to match the content bandwidth, the media coding and rendering complexity to the available network and terminal resources, while maximizing the overall perceived quality. The process of matching the complexity of multimedia content to terminal resources is often referred to as Terminal Quality of Service (QoS) management.

Our previous work [1] and that of others in the QoS community [2] presented QoS management frameworks for adapting (reconfiguring) the (rendering of) scalable applications to platform resources on microprocessor platforms. With the current evolution in run-time reconfigurable computing techniques and field programmable device technologies [3] [4], it also becomes possible to match the platform to the rendering task to be performed. The Terminal QoS manager for a hybrid processor(s)/FPGA platform could e.g., decide to configure the FPGA as a MPEG-2 decoder when the main task is video decoding or configure the FPGA as a 3D pipeline when mostly 3D objects have to be shown (remaining tasks are then run on the processor(s)).

The paper is organised as follows. Section 2 presents the application model for scalable multimedia applications. Section 3 describes a QoS management middleware for run-time reconfigurable platforms. Section 4 formulates the QoS adaptation problem in terms of an optimisation problem. Finally, section 5 proposes a genetic algorithm to solve the optimisation problem.

2. APPLICATION MODEL

Consider one application which has M different objects O_1, O_2,...,O_M sharing N tasks T_1-T_N, each object has L_k quality levels. Each quality level j of object k corresponds to a benefit value b_{kj} which represents the degree of user satisfaction when receiving this quality level. Let w_k denote the relative importance of object k.

Let us further define:

- For each object k, L_k*N mapping matrix α^k that maps quality levels on tasks: α^k_{jt}=1 if task T_t is needed in order to obtain quality level j, α^k_{jt}=0 otherwise.
- For each task T_i, a variable β_i, β_i =1 if T_i is implemented in HW, β_i =0 if T_i is implemented in SW.
- $t_{Ti}(L_{kj}, \beta_i)$ is the execution time of task T_i when processing object k at quality j with implementation β_i. This time is the sum of the actual execution time and the communication time to send the data from this task to other tasks or to the environment.
- $M_{Ti}(L_{kj}, \beta_i)$ is the memory needed by task T_i when processing object k at quality j with implementation β_i.
- $P_{Ti}(L_{kj}, \beta_i)$ is the power consumption of task T_i when processing object k at quality j with implementation β_i.
- $A_{Ti}(L_{kj}, \beta_i)$ is the hardware area needed by task T_i when processing object k at quality j with implementation β_i. Therefore $A_{Ti}(L_{kj}, 0)$= 0.
- $B(L_{kj}, T_l, T_m)$ is the data unit to be communicated between task l and m when processing object k at quality j.
- $B(L_{kj}, T_l, env)$ is the data unit to be communicated between task l and the environment when processing object k at quality j.

3. QOS MIDDLEWARE

Figure 1 shows the QoS middleware, interfacing with the application and the SW/HW multitasking operating system. This middleware consists out of: (i) a QoS based SW/HW partitioner (QHSP); (ii) a resource estimator (RE); and (iii) a SW/HW translator (HST). The main task of the QoS based SW/HW partitioner is to decide which tasks should be put in SW, which tasks should be put in HW and at which quality level in such a way that the user receives the highest possible quality level of the application given the resource constraints (e.g. hardware area). The resource estimator is in charge of estimating the resource requirements for each task based on high-level information from the application (e.g., frame rate, resolution, metadata etc.) and information specific to the platform (which typically can be obtained by profiling). These resource requirements are input for the QHSP to make the partitioning decision. The HW/SW translator translates the platform independent SW and HW code to platform specific code [5]. The translated SW code will run on the processor while the translated HW code will be used to configure the FPGA.

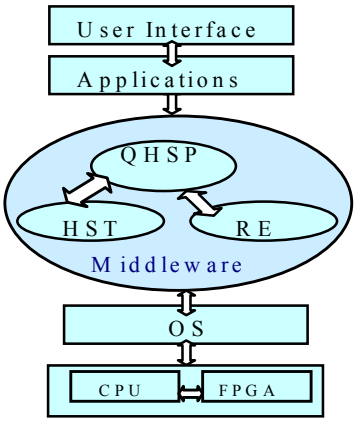

Figure 1. QoS Middleware on reconfigurable platforms

4. FORMULATION OF QOS BASED HW/SW PARTITIONING PROBLEM

The objective of the QoS based HW/SW partitioning is to find a partitioning solution in such a way that the overall quality of the application is maximised under resource constraints. Mathematically, the problem can be formulated as follows.

Maximize:

$$B = \sum_{k=1}^{M} w_k \sum_{j=1}^{L_k} b_{kj}.x_{kj}$$

where $\sum_{j=1}^{L_k} x_{kj} = 1, k = 0,1,...,M$, x_{kj}=1 if quality j is selected, 0 otherwise.

Subject to:

$$T = \sum_{k=1}^{M} \sum_{i=1}^{N} \sum_{j=1}^{L_k} x_{kj}.t_{T_i}(L_{kj}, \beta_i) < T_{\max}$$

$$A = \max_{k=1 to M} \{\sum_{i=1}^{N} \sum_{j=1}^{L_k} x_{kj}.A_{Ti}(L_{kj}, \beta_i)\} < A_{\max}$$

Although only time and hardware area constraints are considered here, other constraints like memory and power consumption can easily be added. B is the application benefit, calculated as the weighted sum of benefit of all objects. T is the total execution time of all tasks for all objects when tasks are executed sequentially. A is the total hardware area needed for the object which requires the most hardware area. T and A are estimated by the RE.

5. A GENETIC ALGORITHM SOLUTION FOR THE PROBLEM

The problem presented in section 4 is a NP-hard optimization problem for which the optimal solution is hard to find at run-time. This section presents how an approximation solution for the problem can be found using a genetic algorithm (GA). Since the GA is under implementation, the evaluation of the algorithm is subject to future work. The rest of the section thus concentrates on modeling the problem through the GA.

A genetic algorithm consists of an iterative procedure during which a series of generations of populations, one per interaction, are created [6]. Each member of a population, called chromosome, represents a solution of the problem being solved. The first generation of populations is often generated randomly. The next generations are created from the first generation using a set of evolution operators including selection, mutation and crossover. Selection operator selects probabilistically the most highly fit members of the current population to move to the next generation. Mutation attempts to introduce new features into the population that did not exist in the previous generation by randomly altering the structures of chromosomes. Crossover operator attempts to combine the feature of highly fit members of a population in a hope of creating new members that are likely to be better fit than either of their parents.

Chromosome encoding: each chromosome is represented by an integer array of size M+N as follows: $\{(q_1, q_2,.....,q_M) (\beta_1, \beta_2, ..., \beta_N)\}$ where q_i represents the quality level of object i and β_j represents the mapping solution for task j. Therefore x_{kj}=1 if j=q_k.

Fitness function: the fitness function is defined as $f = \sum_{k=1}^{M} w_k \sum_{j=1}^{L_k} b_{kj}.x_{kj} + \frac{1}{1+\text{cost}}$

Where cost is defined as: $\text{cost} = \frac{\Delta T}{T_{\max}} + \frac{\Delta A}{A_{\max}}$

where, $\Delta T = \begin{cases} 0 \ \ if \ T \le T_{\max} \\ T - T_{\max} \ \ otherwise \end{cases}$ $\Delta A = \begin{cases} 0 \ \ if \ A \le A_{\max} \\ A - A_{\max} \ \ otherwise \end{cases}$

The fitness function is used to calculate the fitness of each chromosome. The chromosome with the highest fitness value and cost zero in the last generation will be selected as the solution for the problem.

REFERENCES

[1] N. PhamNgoc, W. van Raemdonck, G. Lafruit, G. Deconinck, and R. Lauwereins, "A QoS Framework for Interactive 3D Applications", *Proc. of 10-th Int. Conf. in Central Europe on Computer Graphics, Visualization and Computer Vision'2002*, pp. 317-325, 2002.

[2] Hafid, G.Bochmann and R. Dessouli, "QoS and Distributed Multimedia Applications: A Review", *The Electronic Journal on Networks and Distributed Processing*, issue 6, February 1998.

[3] http://www.xilinx.com/xlnx/xil_prodcat_landingpage.jsp?title=Virtex-II+Pro+FPGAs , 2002.

[4] J-Y. Mignolet, S. Vernalde, D. Verkest, R. Lauwereins, "Enabling Hardware-software Multitasking on a Reconfigurable Computing Platform for Networked Portable Multimedia Appliances," *Proc. of the Int. Conf. on Engineering of Reconfigurable Systems and Algorithms (ERSA)*, pp.116-122, Las Vegas, June 2002.

[5] Y. Ha, S. Vernalde, P. Schaumont, M. Engels, R. Lauwereins and H. De Man "Building a Virtual Framework for Networked Reconfigurable Hardware and Software Objects", *Journal of Supercomputing (*accepted).

[6] Z. Machalewicz, "*Genetic Algorithms + Data Structures = Evolution Programs*", Springer, Berlin, 1999.