RAPID PROTOTYPING OF AN ADAPTIVE NOISE CANCELER USING GRAPE

Luc De Coster, Rudy Lauwereins, J.A. Peperstraete

Katholieke Universiteit Leuven, Department ESAT
Kardinaal Mercierlaan 94, B-3001 Heverlee, Belgium
Tel.: +32-16-32 18 19, Fax: +32-16-32 19 86
email: luc.decoster@esat.kuleuven.ac.be

ABSTRACT

In this paper we describe the rapid prototyping of an adaptive noise canceler using the Graphical RApid Prototyping Environment (GRAPE), developed at our laboratory. It is an environment which facilitates the real-time emulation and implementation of synchronous DSP applications on heterogeneous target platforms consisting of DSPs and FPGAs. Our work demonstrates the feasibility of real-time prototyping using a multiprocessor and a powerful programming framework.

1. INTRODUCTION

The ever increasing complexity and data rates of DSP applications often demand application-specific ICs and hardware. Development costs for such hardware and ASICs are high, so algorithms should be thoroughly tested and optimised before implementation at all design stages. Nowadays, most tests and optimisations are performed by analysis and simulation tools on workstations. Application prototypes are worked out only during the last design stage. However, if we prototype in the earlier stages of the design, we can

- test more parameters in a shorter time
- optimise parameters under real-time conditions
- evaluate an algorithm's subjective qualities, e.g. by listening to the actual result of the algorithm
- prove the feasibility of a design's implementation

Despite these benefits, prototyping is generally not used at the earlier design stages because designing dedicated prototyping hardware is time consuming and expensive.
We propose an alternative: a rapid-prototyping set-up with general-purpose hardware to minimise the development cost and advanced programming tools to reduce the programming time. The general-purpose hardware consists of commercial DSP processors and FPGAs linked together to form a powerful, heterogeneous multiprocessor. Our prototyping environment GRAPE [1] permits easy programming, compiling, simulation, debugging, and testing of real-time DSP algorithms on the multiprocessors.
In this paper we demonstrate the feasibility of this approach with a real-live example: the adaptive noise canceler. In section II we briefly describe the adaptive noise canceler algorithm. Section III describes the implementation of the algorithm, using the GRAPE prototyping environment. Section IV describes the set-up for the demonstration. Finally we draw some conclusions.

2. ADAPTIVE NOISE CANCELLING

Adaptive noise canceling is a basic application in the field of digital signal processing [2]. A signal corrupted by noise is filtered on the basis of a reference of that noise. Car telephones fit in this model. The speaker's signal is so heavily corrupted by the noise caused by engine and tires that it is not understandable anymore. Since the noise source changes within time (e.g. different velocity, different type of road) an adaptive filtering is required. The filter adjusts itself automatically (i.e. filter coefficients). Figure 1 sketches the adaptive noise canceler.

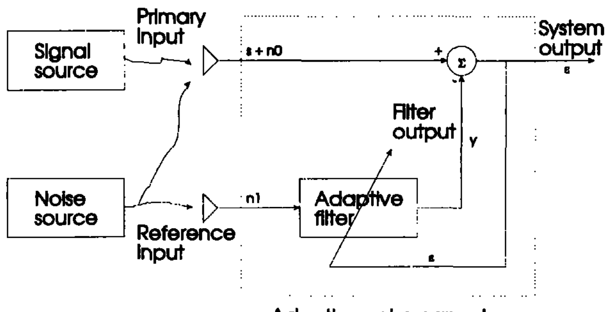

Figure 1: Adaptive noise-cancelling concept.

3. RAPID PROTOTYPING OF AN ADAPTIVE NOISE CANCELER USING GRAPE-II

In the following steps we briefly describe the different phases of GRAPE's design flow.

3.1. Specification of Application and Target

In the specification phase, the application is described using an extended data flow model, called cyclo-static data flow (CSDF) [3]. In short, the application is represented as a directed graph G=(N,E), where the nodes N represent computation tasks, and the edges E the communication of the results (called *tokens*) from a producing to a consuming task. The functionality of the nodes is specified in a conventional high level language like C or VHDL. The number of tokens a task produces respectively consumes during an execution phase of a task is known at compile time, allowing for a compile time analysis of the graph in the next phases of GRAPE's design flow and leading to highly efficient run-time code. The adaptive noise canceler algorithm is split in following computation tasks: two first-order high-pass filters for removing DC values on both inputs, two delay lines for allowing to align both inputs, the actual filter — hierarchically specified by two subfilters to increase inherent parallelism — and finally some 'glue' tasks (i.e. duplication tasks and addition/subtraction tasks). Still in GRAPE's specification phase, the target architecture is specified as a connectivity graph, with an indication of the amount and type of resources each processing device possesses. We used a general-purpose, commercially available multiprocessor with two TMS320C40 DSP processors as target hardware. The concerning board is plugged in a host PC, running the GRAPE environment.

3.2. Verification via Simulation

After the application has been specified, it should be verified for functional correctness. Therefore GRAPE provides a simulation path. A single-processor program is automatically generated which can be compiled and executed on a workstation or PC. The input stimuli are read from file during simulation, and the outputs are written to files. In this way simple programming bugs can be detected in an early stage.

3.3. Performance Estimation

In the third phase, the amount of resources required by each of the tasks when executed on each of the processing devices, is estimated. A tuple — consisting of the execution time and the memory usage — is determined for each such combination. For example, a delay task with a maximal delay of 100 samples on a TMS320C40 takes 26 cycles and 128 words.

3.4. Mapping of Application on Target

Next, the application is mapped onto the target hardware. In this phase, each task is assigned to a specific processing device. Then a communication path is established automatically and transparent for each edge in the application's graph. Finally a compile-time schedule order is determined for each device that minimises the total makespan (i.e. latency) while meeting the other constraints (i.e. memory usage) (Figure 2).

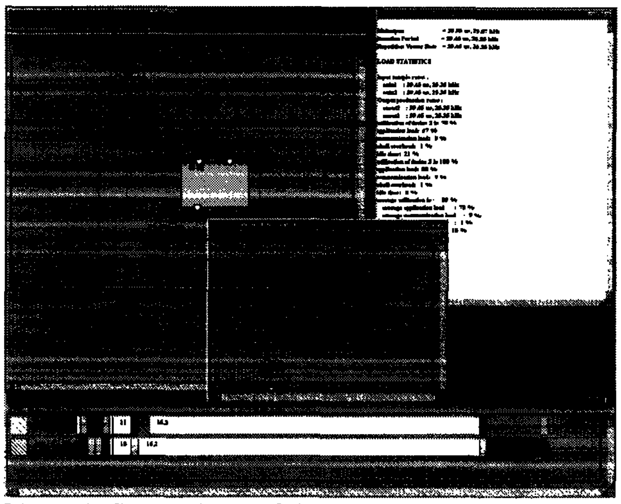
Figure 2: Screen dump of the scheduler. The Gantt chart visualises the schedule of the application tasks and the communication tasks on the two C40 devices. In a performance window the numerical feedback is given.

3.5. Code Generation

In GRAPE's fifth design phase, code in C is generated for each of the processing devices. This phase of 'gluing' gets much attention currently. For example, it considers whether to inline or call the tasks from the main loop and it invokes specific and optimal buffer implementations.

3.6. Real-time Emulation

The final phase is the loading and the running of the application on the target hardware, i.e. the real-time emulation (Figure 3). A major feature in the GRAPE environment is the availability of run-time parameter modification. Following six design parameters for the canceler application can be tuned at this stage: the filter length, the adaptation constant, the lengths of the delays and the cut-off frequency of the HP filters. But also 'back' parameters to the host PC are possible. Snapshots of the filter coefficients allow for objective information (i.e. the impulse response of the filter), next to the subjective experience of the audio output.

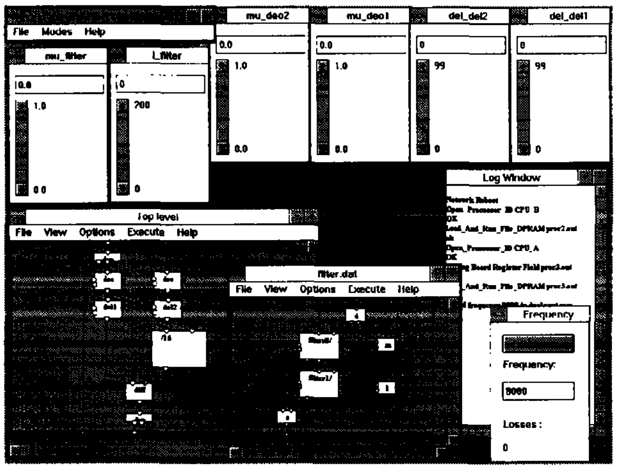

Figure 3: Screen dump of the loader. A hierarchically specified application is mapped on a DSP multiprocessor board. By means of slider bars on the PC, parameters of the application can be tuned at real time. Sample rate can be set and synchronity is monitored in the losses field.

4. DEMONSTRATION

Figure 4 gives the set-up of the demonstration. For ease of presenting, a real situation was recorded once. In the demo the situation is played back and processed at real time. The real situation consisted of a room in which two speakers — one for the speech signal and one for the noise signal — and a microphone where placed. The reference signal for the noise was the noise signal itself. So the adaptive filter should correspond with the concatenations of the transfer functions of speaker, room and microphone.

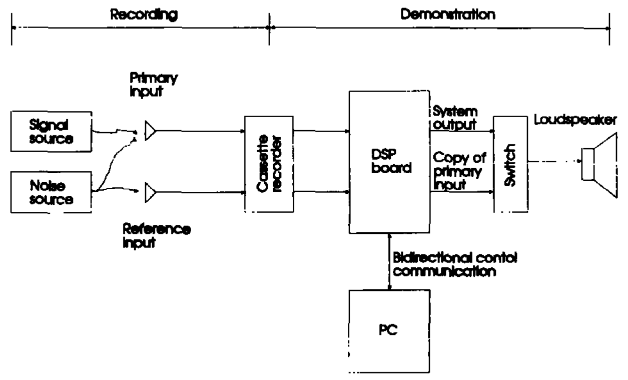

Figure 4: Set-up of the demonstration. For ease of presenting, a real situation was recorded once. In the demo, the situation is played back and processed at real-time.

The demo consist in playing back the recorded signals, feeding them to the application and comparing the result with the original signal. The set-up allows to demonstrate the GRAPE environment. First on overview on the different phases in the design are presented on the PC. Thereafter the real-time emulation is shown. Parameters are changed; snapshots of the filter coefficients are taken and presented by figures on the PC; while listening to the audio result gives the subjective feedback.
Practically, we only need a table to put the demo system on. Since the noise produced is disturbing, a separate room is advisable.

5. CONCLUSION

In this paper we showed the feasibility of our rapid prototyping strategy. It turns out the main benefit of this approach is the support for automatic routing and automatic code generation for the multiprocessor. Also the run-time parameter modification is essential for the tuning of the application. It realises much gain compared with time-consuming simulations on workstations.

Other prototype realisations with GRAPE can be found in [4],[5],[6],[7].

REFERENCES

[1] R. Lauwereins, M. Engels, M. Adé, J.A. Peperstraete, "Grape-II: A System-Level Prototyping Environment for DSP Applications", IEEE Computer, pp. 35-43, Feb. 1995

[2] B. Widrow, S.D. Stearns, "Adaptive Signal Processing", Prentice-Hall, 1985

[3] G. Bilsen, M. Engels, R. Lauwereins, J.A. Peperstraete, "Cyclo-Static Dataflow", IEEE Transactions on Signal Processing, Vol. 44, No. 2, pp. 397-408, Feb. 1996

[4] M. Engels, T. Meng, "Rapid Prototyping of a Real-Time Video Encoder", IEEE International Workshop on Rapid System Prototyping, Grenoble, France, pp. 8-15, June 20-23, 1994

[5] E. Isikli, M. Engels, R. Lauwereins, J.A. Peperstraete, "A Rapid Prototyping Example Using GRAPE-II: An Adaptive Antenna Beamformer for GSM Base Stations", IASTED International Conference on signal and image processing and applications, Annecy, France, pp. 110-113, June 12-14, 1996

[6] P. Wauters, S. Van Gerven, M. Engels, R. Lauwereins, J.A. Peperstraete, "Rapid Prototyping of an adaptive Speech Beamformer using GRAPE-II", International Conference on Signal Processing Applications & Technology, Boston, MA, USA, pp. 1678-1682, Oct. 24-26, 1995

[7] R. Lauwereins, M. Adé, P. Vandaele, M. Moonen, "Prototyping Quadrature Amplitude Modulation for Two-way Communication on CATV Networks", International Conference on Signal Processing Applications & Technology, Boston, MA, USA, pp. 1570-1574, Oct. 8-10, 1996

ACKNOWLEDGEMENTS

Luc De Coster is a Research Assistant and Rudy Lauwereins a Senior Research Associate of the Fund for Scientific Research - Flanders (Belgium). The GRAPE project is partly sponsored by the Belgian Interuniversity Pole of Attraction IUAP-50, 4/20, 4/24, by an NFWO Krediet aan Navorsers, by the Esprit OMI DISSAP-II project 8068, by the European Space Agency project on Multi-DSP architectures, by the Texas Instruments Elite University Program, and by the IWT Information Technology project Isis. K.U.Leuven-ESAT is a member of the DSP Valley ™ network.