Prototyping Quadrature Amplitude Modulation for Two-way Communication on CATV Networks

R. Lauwereins, M. Adé
KULeuven-ESAT/ACCA
Kard. Mercierlaan 94
B-3001 Heverlee, Belgium

P. Vandaele, M. Moonen
KULeuven-ESAT/SISTA
Kard. Mercierlaan 94
B-3001 Heverlee, Belgium

P. Schaumont
Imec-VSDM/DISTA
Kapeldreef 75
B-3001 Heverlee, Belgium

Email: Rudy.Lauwereins@esat.kuleuven.ac.be

1. Abstract

The recently discovered potential of two-way communication on CATV networks for advanced telecommunications applications like video-on-demand, spawned research and development in modem design for up-stream communication. This paper reports on the prototyping of such a 16QAM modem and compares the achievable sample rates on 4 DSP processors to simulation speeds obtained on a powerful workstation.

2. Introduction

Recently, coaxial cable networks have received much attention in the context of interactive application [1,2]. In countries with a high penetration of CATV (e.g. Belgium >90%), the cable network forms a viable alternative to classical telephone networks. Envisaged applications are telephony, interactive television, home-shopping, video-on-demand, high-speed Web browsing, etc. The interactive nature of these services requires however two-way communication on a network that initially was only intended for a one-way broadcasting of television signals. What is aimed at now, is a low bit-rate upstream link (from the subscriber to the head-end station) and a high bit-rate downstream link (from the head-end station to the local subscribers). Particularly the first problem is challenging because very little is known about the upstream channel, and communication standards are still under development [3]. The projected frequency band for upstream communication is in the 5-25 MHz range, see also Figure 1.

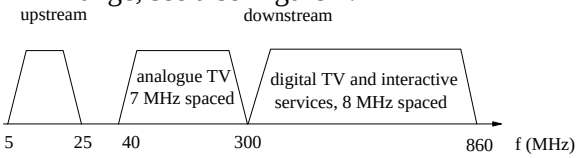

Figure 1. Frequency allocation

The basic configuration of the network is hybrid fibre-coax. This means that the upstream signal will first travel through coax before entering a fibre node and going through the fibre trunk to the head-end station. The coaxial part will bring along some serious channel impairments [4] which must be compensated for at the receiver. Without going into too much detail we only mention the most important impairments. The group delay distortion (i.e. signals at different frequencies propagate with a different velocity through the network) causes severe inter-symbol interference at the receiver. Micro-reflections are caused by discontinuities in the transmission medium and cause part of the signal energy to be reflected. Ingress noise models the interference caused by the antenna like properties of the cable. Burst noise typically originates from household appliances such as electrical motors. Besides this there are common path distortion products, thermal noise, impulse noise, non-linearities, phase-noise frequency offset etc. If we add to this the variations in the networks stemming from the variability of the number of trunk, bridge and distribution amplifiers it becomes quite clear that it is very hard to build a channel model which

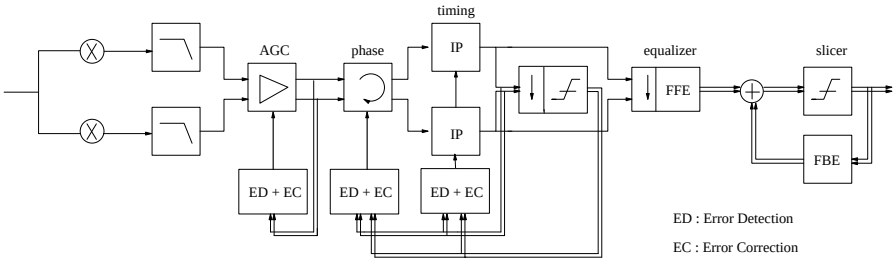

Figure 2. Receiver architecture

incorporates all these statistical and non-statistical phenomena observed in real networks. Studies [2] have shown that because of this ugly environment, in many systems, less than half of the spectrum will be available at any given time instant. The absence of a good channel model necessitates the real-time prototyping of the complete set-up, including the cable, after off-line simulation and before a commitment to silicon is made. Real-time prototyping has several benefits compared to off-line simulation:

- It enables algorithm verification on the real channel;
- It allows for more extensive testing under more varying conditions: 24 hours a day measurements are possible as well as tests with transmitters placed on different locations;
- It gives faster feedback when modifying algorithmic settings: the prototype indeed allows to modify algorithmic parameters on the fly, without re-compilation and to view its effects in real-time on the next received data packet.

We aim at developing a 16QAM modem for this upstream communication channel. The projected bit rate is 10 Mbit per second and this at a bit-error rate of 10^{-10}. The transmission payload consists of ATM cells and the multiple access protocol is TDMA. This type of multi-access protocol naturally fits the cell-based payload.

The next section explains the receiver architecture. Section 4 describes the rapid prototyping environment GRAPE and indicates how GRAPE was used to obtain first estimates of the achievable sample rate when the 16QAM receiver is implemented on 4 Digital Signal Processors (the TMS320C40).

3. The receiver architecture

The receiver structure ought to be the best compromise between low bit-error rate, short run-in sequences (this is particularly important because a burst mode system becomes very inefficient for long run-in sequences), possibility for digital integration and implementation cost. The solution has to be robust against the various channel impairments and should be able to cope with high dynamic ranges (30 dB). The general structure of the receiver is depicted in Figure 2.

Matched filtering is done by means of a fixed root raised cosine filter. Then the Automatic Gain Control brings the signal back to the right level in order to avoid under- or overflow in subsequent sections of the demodulator. A phase loop compensates phase mismatches (which result in a rotation of the constellation diagram) as well as mismatches between the carrier frequency of transmitter and receiver. Timing recovery is done by means of interpolation. After the timing recovery the signal is down-sampled to symbol rate and finally the channel distortion is countered using an equaliser. Since the compensation of the group delay distortion is the only task of the equaliser, it converges quite rapidly, even with a slowly converging least mean squares (LMS) algorithm. Performance was improved using a decision feedback equaliser (DFE), where the decisions of the slicer are fed into the feedback part of the equaliser. If the decisions of the slicer are correct, the input of the feedback part is error free and hence improves performance. Since the output of the slicer is only a few bits wide, a very cost effective implementation is possible. The equaliser is de-coupled from the rest of the receiver structure by means of the first slicer, this in order to avoid loop instability caused by

interference. The error correcting mechanisms work with different time constants during training and tracking phase. During training a fast acquisition is desirable while in tracking only slow variations of the channel characteristics have to be compensated.

4. Prototyping with GRAPE

This section first describes the design flow of the prototyping environment GRAPE, developed at the K.U.Leuven. In the next section, it explains how GRAPE is used to prototype the 16QAM receiver and what sample rates may be expected on a target consisting of four TMS320C40 DSP processors.

4.1. GRAPE's design flow

GRAPE (Graphical RApid Prototyping Environment) is an environment, developed at our laboratory, which facilitates the real-time emulation and implementation of synchronous DSP applications on heterogeneous target platforms consisting of DSPs and FPGAs [5]. Many aspects of GRAPE resemble the environments Ptolemy of UC Berkeley [6] and COSSAP of RWTH Aachen [7], currently further developed by Synopsys; the main distinction is that GRAPE is targeted at real-time execution whereas the other environments mainly target simulation.

GRAPE's design flow consists of four phases. In the specification phase, the application is described using an extended data flow model, called cyclo-static data flow (CSDF) [8], which is an extension of Lee's Synchronous Data Flow [9]. 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 and 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. 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 [10]. In the second phase, the amount of resources required by each of the tasks when executed on each of the processing devices, is estimated. Next, the application is mapped onto the target hardware. In this phase, each task is assigned to a specific processing device, a communication path is established for each edge in the application's graph and a compile time schedule order is determined per device that minimises the total makespan. In GRAPE's fourth and last design phase, code in C or VHDL is generated for each of the processing devices, consisting of a main program and communication primitives. Note that a single design flow is used for software targets (DSPs) as well as for hardware targets (FPGAs) [11].

4.2. Prototype of the 16QAM receiver

The target platform available for implementing the prototype of the receiver, consists of 4 fully interconnected TMS320C40 processors, running at 40 MHz. It consists of two PC long ISA-bus cards. Both cards communicate with the host PC via dual ported RAM for program downloading and modification of algorithmic parameters.

First, the 16QAM receiver application is specified using GRAPE's graphical editor. By carefully inspecting Figure 2, we can increase the granularity of the application wherever several sub-tasks are clearly sequential and cannot be pipelined. This reduces the amount of inter-task communication overhead. We end up with Figure 3, which is a screen-dump of the specification tool of GRAPE. The (dark grey) triangles represent algorithmic parameters that may be modified at run-time. We clearly see the matched filters (FIR), the automatic gain control (AGC), the phase loop (PHI), the symbol alignment (MU) and the equaliser (LMS).

The tasks of the 16QAM receiver as specified in Figure 3, have been automatically assigned to the 4-processor target hardware, as shown by the shade of grey of the task borders in the application window of Figure 4.

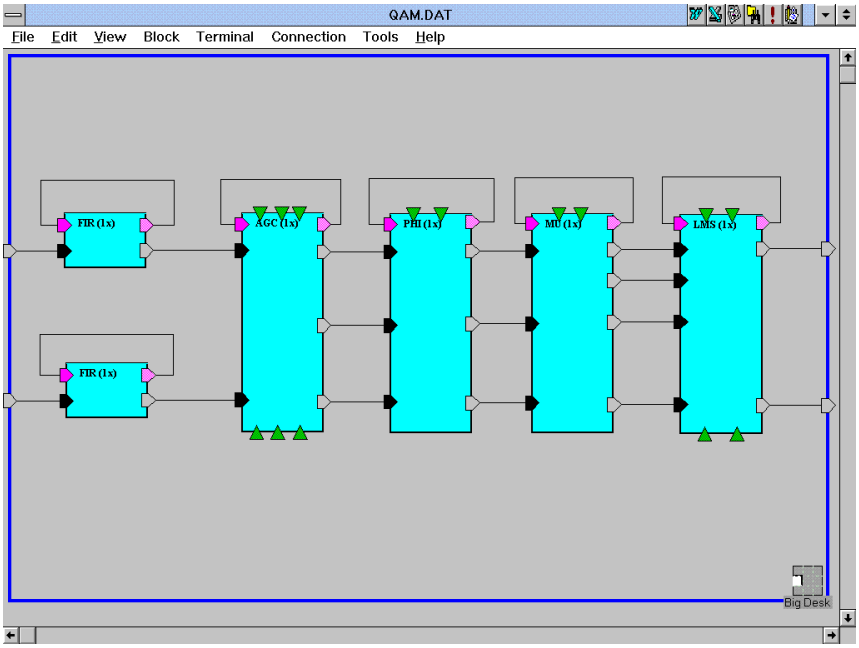

Figure 3. High level specification of the QAM receiver.

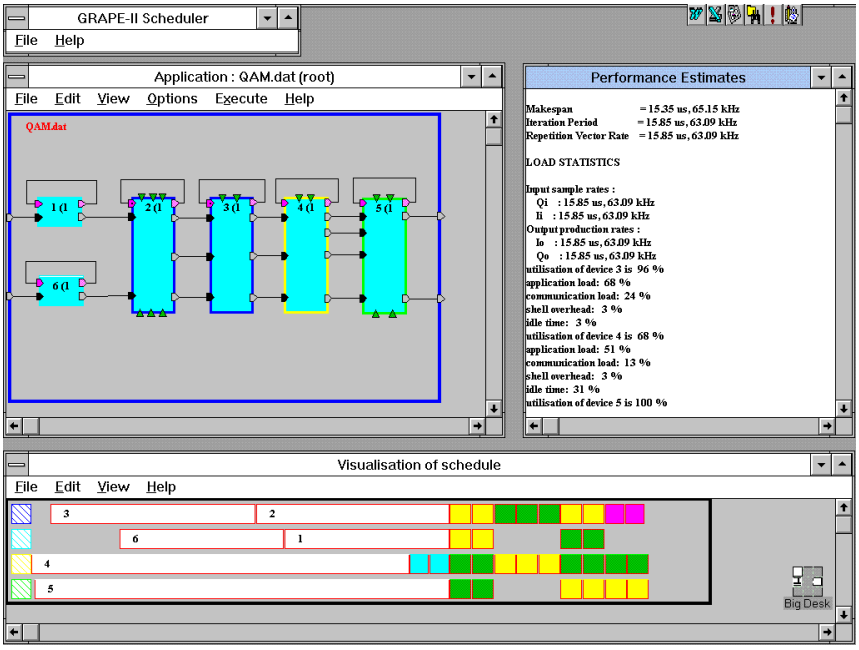

Figure 4. Schedule of the high level tasks.

Then, GRAPE ordered the tasks on each device in time, such that processor idle time is minimised. The bottom of Figure 4 shows this schedule. White tasks are application tasks; grey shaded tasks are inter-device communication primitives, that were automatically generated by GRAPE. The obtained sample rate of 63 kHz corresponds to a symbol rate of 15.770 symbols per second. Table 1 compares the data throughput with simulation and ASIC implementation.

	Symbols/s	Relative to real-time
ASIC	2.500.000	1
Prototyping	15.770	159
Simulation (HP700)	500	5.000

Table 1 Estimated data rates.

As can be seen, real-time prototyping cannot be achieved. However, when the QAM protocol is implemented in burst mode and when more than 159 transmitters are in a time-multiplexed way present on the same CATV cable, real-time processing for one user can be obtained by buffering a complete burst and processing it when the burst of the other users are on the cable.

It is expected that a substantial speed improvement can be obtained compared to the figure indicated above. These will be investigated in the remainder of the project. A limited list of possibilities follows:

1. A very cheap improvement is to switch to 50 MHz processors.
2. The current implementation requires the copying of the results of one block into a software buffer before communication primitives copy them onto the hardware links. Careful scheduling can avoid these copying steps and the associated communication primitives. This would remove all shaded tasks in Figure 4.
3. The current C implementation is not optimized for the TMS320C40 DSP processor. The speed gain is unpredictable. Previous experience showed speed gains between 20% and 300%.
4. Timing critical tasks, especially those containing bit manipulations or extensive conditional processing, may be migrated to FPGAs. Speed gain is unpredictable. This migration reduces the flexibility and observability of the application, and requires a substantial amount of effort, since the C specification of the migrated sub-task needs to be re-written in register transfer level VHDL. This migration will hence only be done during the later stages of prototyping. Previous experience has shown speed gains up to 800%.

5. Conclusion

Although not achieving real-time sampling rates, prototyping is shown to be valuable to evaluate the interaction of new modem designs with the real channel, by offering a speed-up of one to two orders of magnitude compared to workstation simulation. The use of advanced prototyping environments like GRAPE in combination with programmable target hardware makes prototyping hardly more expensive in development time and equipment than simulation.

6. Acknowledgements

R. Lauwereins and M. Moonen are Senior Research Associates with the NFWO. P. Vandaele is a Research Assistant of the IWT. Research supported by Siemens Atea and the Flemish Government via the Flemish Institute for the Advancement of Scientific-Technological Research in Industry (IWT). This project has partly been made possible by NFWO, ESA, Esprit (Retides, Dipsap-II) and Texas Instruments. K.U.Leuven-ESAT and Imec are members of the DSP Valley network.

7. References

1 Comerford R. and Tekla S., “Wired for Interactivity”, *IEEE Spectrum*, April 1996, pp 21-28.

2 Goldberg L., “Cable Modems: The Journey From Hype To Hardware”, *Electronic Design*, Apr. 1996, pp 65-80.

3 Eng W., “IEEE Project 802.14: Standards for Digital Convergence”, *IEEE Communications Magazine*, May 1995, pp 20-23.

4 Currivan B., “CATV Upstream Channel Model, Rev 1.0”, *IEEE P 802.14 Working Group*, June 1996.

7 R. Lauwereins, M. Engels, M. Adé, J.A. Peperstraete, “Grape-II: A System-Level Prototyping Environment for DSP Applications”, *IEEE Computer*, Feb. 1995, pp 35-43.

8 J. Buck, S. Ha, E.A. Lee, D.G. Messerschmitt, “Ptolemy: a Framework for Simulating and Prototyping Heterogeneous Systems”, *Int. Journal of Computer Simulation*, Vol. 4, Apr. 1994, pp 155-182.

9 Synopsys Inc., 700 E. Middlefield Rd., Mountain View, CA 94043, USA, COSSAP User’s Manual.

10 G. Bilsen, M. Engels, R. Lauwereins, J.A. Peperstraete, “Cyclo-Static Dataflow”, *IEEE Trans. on Signal Processing*, Feb. 1996.

11 E.A. Lee, D.G. Messerschmitt, “Static Scheduling of Synchronous Data Flow Programs for Digital Signal Processing”, *IEEE Trans. on Computers*, Vol. C-36, No. 1, Jan. 1987, pp 24-35.

12 G. Bilsen, M. Engels, R. Lauwereins, J.A. Peperstraete, “Compile-time Makespan-optimal Multi-resource Mapping for Hardware/Software Co-design”, *KULeuven Tech. Report ESAT-ACCA 95-02*.

13 M. Adé, R. Lauwereins, J.A. Peperstraete, “Hardware-Software Co-design with GRAPE”, *Proc. 6th Int. Workshop on Rapid System Prototyping*, Chapel Hill, NC, USA, June 1995, pp 40-47.