A technique for combined virtual prototyping and hardware design

Patrick Schaumont Geert Vanmeerbeeck Erik Watzeels Serge Vernalde
Mark Engels Ivo Bolsens

Abstract

A technique to include virtual prototyping in the design cycle of complex digital modem ASICs is presented. It is innovating by using the same behavioral description for both the prototype as well as the final circuit implementation. Relating to verification of the design, this is a crucial benefit. The article discusses the prototyping mechanism by using the design of an upstream cable modem ASIC as a driving example. Also, the importance of prototyping is positioned within the design flow used to develop this cable modem.

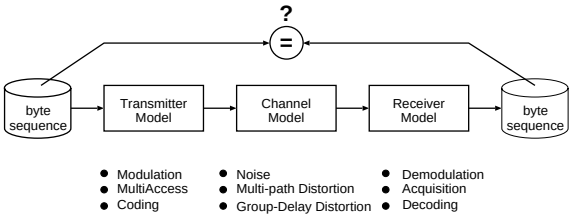

Figure 1. End-to-end model of a communications system

1. Introduction

High-speed digital modems are currently being developed for wire-line access networks, like CATV networks and Digital Subscriber Lines as well as for broadband wireless communications, e.g. wireless LAN, wireless ATM or mobile broadband systems. A common aspect of these modems is that they make use of complex signal processing algorithms, like adaptive equalizers, Viterbi decoders and FFTs, that have to be performed at high speed. To optimize the performance of these modems with respect to bandwidth efficiency and bit error rates, the algorithms must be optimally tuned to the characteristics of the communication channel. For instance, the equalizer for a wireless indoor channel has to combat a large number of multi-path reflections and thus will be completely different from the equalization for a CATV modem where the main function is to cancel group delay distortion.

Therefore, it is common practice to optimize the algorithmic parameters of a modem algorithm by an end-to-end simulation set-up (Figure 1). In such an end-to-end simulation the behavioral description of the modem is co-simulated with a model of the transmission channel. However, these channel models are often not established for new channel types, such as the upstream CATV communications path. Moreover, time-varying propagation, ingress noise, impulsive noise, inter-user interference, etc. are effects which are hard to model accurately. Because the choice of the algorithmic parameters strongly depends on this channel model, these limitations translate in suboptimalities or even errors in the modem algorithm. Such errors force redesign cycles which are expensive in terms of time-to-market and design cost.

The solution to this problem is the construction of a prototype that can be used on the real physical communication channel. The prototype should have sufficient flexibility to operate with a large number of different algorithmic parameters and its development time should be at orders of magnitude shorter than for the final product. In literature, two such rapid prototyping approaches are being advocated.

The first approach is to map the modem algorithm onto a general purpose processor or programmable DSP. With this approach only low to medium rate applications operate in real-time, although the achievable data-rate can be increased by using a multiprocessor. Rapid prototyping environments like GRAPE [4] and Ptolemy [5] aim at automating the construction of these prototypes. However, they pose some restrictions on the computational model to be used, e.g. synchronous data-flow. Most often the prototyping will be performed at the floating point functional level. Bit true simulations are possible but at the cost of a considerable overhead. The major shortcomings of this approach are the missing support for timing true prototypes and the lack of a path towards ASIC implementation.

The second rapid prototyping approach consists of mapping the modem architecture onto one or multiple FPGAs [6, 2]. By nature, bit-true behavior and timing of the archi-

tecture are present in the prototype. Although this approach is very popular for the emulation of large general purpose processors at reduced clock speeds [3], it is less accepted for complex high-speed modem ASICs. The reasons are that these modems typically require multiple FPGAs and real-time operation at a relative high clock rate. As a consequence, such a prototype can only be realized by optimizing the architecture for the FPGA implementation. As a consequence, the timing true emulation capabilities are lost as well as the direct path to ASIC implementation. Also the automatic prototyping tools for FPGAs are less developed, resulting in a considerable design effort for building the prototype.

In this paper we propose an alternative prototyping approach, in which a single C++ software description is used for both the prototype as well as for the final ASIC implementation. This approach is based on data generation and acquisition hardware. Signal bursts are computed off-line and stored in memory and then transmitted over the physical medium in real-time. The received data is captured in memory and the receiver signal processing is done off-line. With this virtual prototyping approach real-time transmission on the real network is possible for burst-mode modems, for instance in wireless cellular communications, and upstream CATV communication. The C++ model can be specified bit true as well as cycle true and does not impose any restrictions in the programming model. Besides the single software description for prototyping and implementation, the virtual prototyping approach offers the advantage of very short prototype development times as well as the usage of simple off-the-shelf hardware.

The remainder of this paper is organized as follows. In the next section an application for this prototyping approach is introduced, which is an upstream CATV cable modem. Next, we will discuss the setup of our C++ design environment in which this application is captured. The fourth section presents the prototyping approach of the cable modem. Finally, conclusions from this work are drawn.

2. Upstream CATV Communications

Figure 2 shows the environment of an upstream CATV network. The cable modem receiver [1], discussed in this article, is located inside the head-end which connects the access network (hybrid fiber-coax) to an ATM core network. The access network part contains active amplifier elements as well as passive wiring and power splitting elements (taps). At the user end, the CATV outlet is plugged into a set-top box, which offers the user a high speed data interface. A full implementation requires bidirectional communication facilities between the user and the head-end. For simplicity, only the communication from user to head-end is discussed.

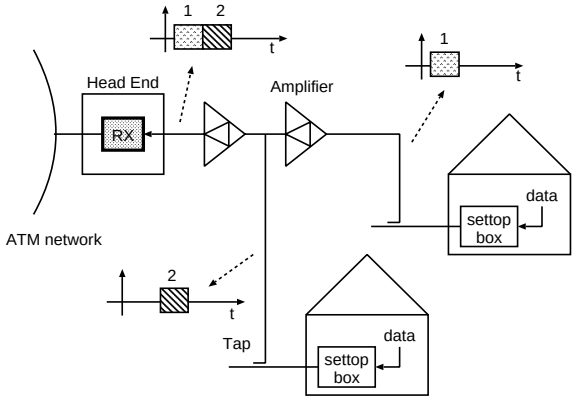

Figure 2. Upstream CATV network

The communications path between the users and the head-end is a tree network, and the users must share the upstream bandwidth amongst each other. Current standards, as well as this work, are in favor of a time division multiple access (TDMA) protocol between the different users. This means that each user in turn gets access to the full upstream bandwidth. The entire upstream channel spans 5 MHz to 42 (USA) or 65 (Europe) MHz, which is subdivided into different TDMA channels. Our application requires a TDMA channel of 3 MHz, onto which a QAM16 modulation scheme is applied. This yields a bit-rate of 10 Mbit/s, to be subdivided amongst the users of one TDMA channel.

The modulations generated by the users are bounded by one timeslot of the TDMA protocol, which results in burst-type signals that travel along the upstream CATV path to the head-end. The timeslots are sufficiently long to contain one ATM-cell plus some overhead necessary for the modulation scheme.

This scenario results in the following challenges for the receiver located in the headend:

- The receiver needs to demodulate the burst-type signals within one timeslot, since each timeslot can be allocated to an active user.
- The receiver needs to estimate all modulation parameters on a per-burst basis because each burst can arrive from a different source. The modulation parameters include signal level, symbol timing and carrier phase (quadrature demodulation needs a synchronous carrier reference).
- The required data filtering at the receiver varies on a per burst basis. The pulse shape of the received signal suffers from delay distortion introduced by the active elements in the upstream CATV path. Failure to take this into account causes inter symbol interference (ISI) at the receiver data filter output, and hence degrades bit-error rate performance.

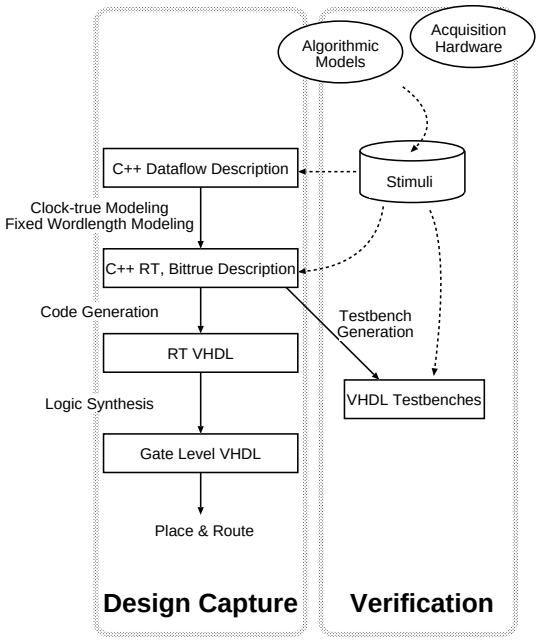

Figure 3. Integrated Protoyping/ASIC Design Flow

For this receiver, a reception and demodulation algorithm was developed in cooperation with the SISTA group of ESAT-KUL. The algorithm uses advanced digital signal processing and is computational intensive: the digital hardware implementation of the algorithm has a complexity of 80 Kgates.

The throughput and complexity of data processing required for the receiver introduces an additional requirement, namely that an integrated, application-specific solution (ASIC) is needed. ASICs, however, need relatively long development cycles and design iterations are very costly. Therefore, a prototype is required. If the prototype and the ASIC are realized sequentially, the development cycle would be further extended. Therefore we propose a strategy where ASIC development and prototyping are performed concurrently. This requires a development environment that integrates the ASIC prototyping and design methodology very closely.

3. Development Environment

The development environment, shown in figure 3, has two major branches. The first is related to design capture, which is the encoding of the cable modems' behavior at different abstractiong levels. The second branch contains the verification strategy, needed to check the functional correctness of one abstraction level as well as to check the consistency amongst different abstraction levels.

The design capture hierarchy spans four different abstraction levels.

1. The top level is encoded in C++ and uses a data-flow simulation paradigm. The upstream cable modem demodulation algorithm is splitted in different sub-functions, each of which are mapped to one actor.

2. The next level is a bit-true, cycle true model of the modem in C++. It is obtained by scheduling the algorithms out of the data-flow actors (cycle true modeling), and by quantizing the variables used by these to a fixed point representation. Scheduling and quantization proceed as to minimize the hardware resources needed by the ASIC implementation.

3. The following level is a synthesizable, register transfer (RT) level representation of the modem behavior in VHDL. This level is semantically equivalent to the previous one, and we use an automatic translation approach to obtain the VHDL out of the C++ description.

4. The lowest level is a technology mapped, gate level VHDL netlist. It is generated out of the RT-VHDL by logic synthesis.

The different design capture levels use stimuli designed in the verification branch of the flow. Sources for stimuli are algorithmic models of the upstream channel environment, and acquisition hardware. The acquisition hardware allows to capture real-life signals from a CATV network, which are free from inaccuracies or ambiguities in the algorithmic channel model.

The stimuli can be directly read in by the C++ coding levels. To use these at the lower coding levels in VHDL, a testbench is automatically generated out of the bit-true, clock-cycle true C++.

To support the different modeling aspects in C++ (data-flow, fixed point quantization, cycle-true, VHDL code generation) we have developed a generic C++ library that can be reused in different designs. The presence of acquisition hardware, which is steered out of the C++ ASIC design environment, allows a tight integration of prototyping and design activities. In the next section, we will focus on the hardware and software prototyping setup for the case of the cable modem.

4. Prototyping Setup

4.1. Overall System

The system setup used for the development of the cable modem is shown in figure 4. Although only the receiver

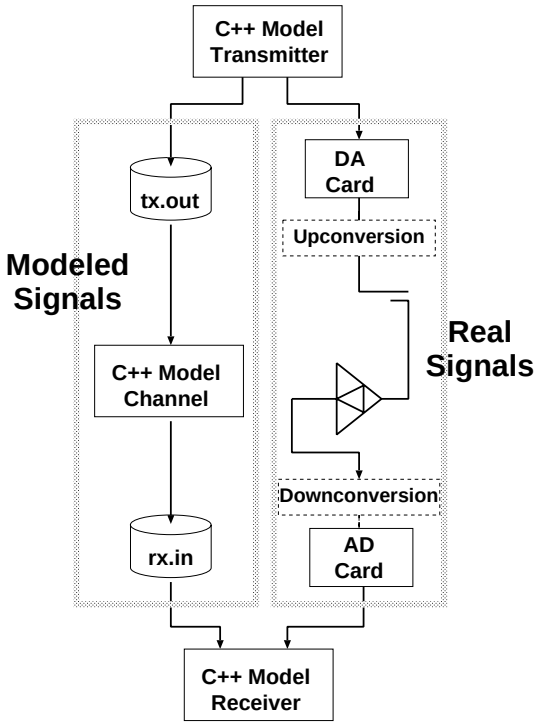

Figure 4. Development System Setup

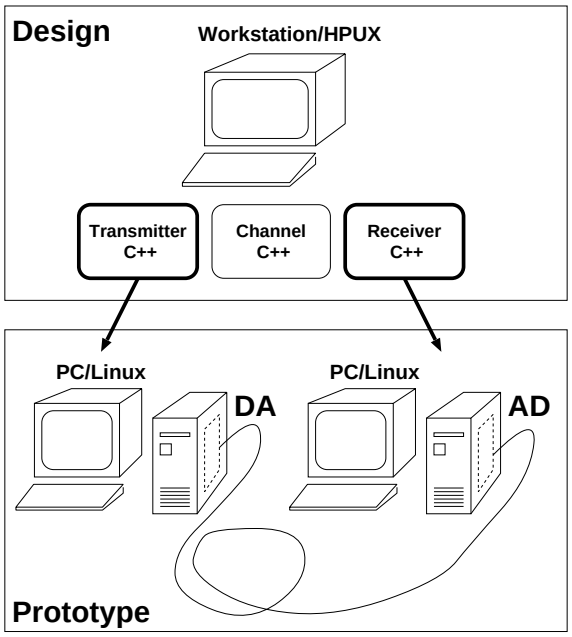

Figure 5. Development Hardware

needs to be designed in real hardware, it is seen that an end-to-end model was developed, including the transmitter, the channel, and the receiver.

The transmitter is a C++ program using the data-flow semantics. The purpose is to generate burst-type QAM16 signals out of a user-specified byte sequence.

The channel can be either a software model in C++ (data-flow), or else the physical upstream CATV channel itself. In the latter case, a burst signal generated by the transmitter model is stored in the memory buffer of a digital-to-analog conversion card (DA). The DA card then scans out the buffer to generate a repeating burst signal on the physical channel. The transmission on the actual upstream CATV channel also requires analog frequency translations, which are inserted behind the DA and before the AD card.

At the receiver side, the burst signal is acquired in a file (for the software channel model), or else by an analog-to-digital conversion card. The power of the repeating burst signal triggers the acquisition process on the AD card. Once triggered, the card fills a frame buffer sufficiently long to store at least one complete burst signal. Either received signal can then be processed by the receiver C++ description. Depending on the design stage the high level data-flow description or else the clock-cycle true, bit-true description is used.

4.2. Hardware

The hardware platform for the modem ASIC development is shown in figure 5. For the design, a workstation platform is needed, which offers the computational power needed for typical ASIC related design tasks such as logic synthesis. For prototyping, two personal computers are used, which run the C++ description of the transmitter and the receiver respectively. Also, the PC contains a DA card (transmitter PC) or an AD card (receiver PC).

The cost of the prototyping platform is primarily determined by the cost of the data acquisition hardware: we used a 200 MHz, 12 bit Signatec DA card and a 100 MHz, 12 bit Signatec AD card. This bandwidth is necessary to adequately represent the wide-band signals on the channel. It offers however no real time operation of the receiver modem: once a burst signal is captured by the AD card, the relatively slow demodulation process is run off-line.

The advantages of this prototyping approach can be summarized as follows.

- The prototyping environment runs a compatible operating system as the design environment (Linux as opposed to HP/UX). The C++ code was written under GNU g++ and requires no porting effort when moving code between design and prototyping platforms. As will be shown in the next section, the impact of changing software stimuli in files with stimuli gathered on the AD and DA card is kept minimal.
- The largest unknown factor in the design of a modem

is the channel. More important then processing speed, the processing of real-life captured burst signals allow to check the algorithms at least functionally.

- The separation of transmitter and receiver introduces asynchrony in the modem model. This allows to verify the correct operation of symbol timing recovery algorithms, which is much more difficult to do in an all-software environment. In addition, a separated end-to-end model allows experiments on remote environments.

- The porting of the ASIC model itself, in unaltered form, allows to verify the correctness of the C++ cycle-true, bit-true behavior. In case accelerator hardware would have been used, an additional equivalence would have to be proved between the accelerated model and the ASIC model.

On the downside, we indicate the following drawbacks

- Because of the low processing speed of the prototype, extended simulation runs are infeasible. Consequently, low bit error rate performance cannot be measured adequately with the prototype.

- The prototype is a feed-forward model. Connecting for instance two different users (transmitters) to the same receiver is not possible since this would require real-time arbitration of a TDMA protocol. In general, real-time interactions with the transmission protocol cannot be simulated.

In conclusion, for real-time bit-error rate performance and protocol interactions, accelerator hardware will be needed. To develop a functionally correct ASIC however, the prototype is adequate.

4.3. Merging Acquisition Hardware into the Design Environment

Since the acquisition hardware is part of the ASIC verification environment, a close software integration in the design system is needed. We solved this by an approach as shown in figure 6.

The figure shows the case of the AD card. This card is equipped with a PCI interface for fast transfer of acquired data to the application. To hook up this card into the Linux operating system, we wrote a device driver `driver.c`, which accesses the AD hardware and transfers it to the file system. The development of the driver was completed within two weeks.

As seen from the file system, the control functions of the card and the data transfer path are available within one logical channel. Once the driver is inserted into the operating system, an application can access it as it were a normal file like one on a hard-disk.

This file is accessed from within a data-flow actor from the design environment. Two data-flow actors are available for file input. The first one, `source.cxx`, reads a plain file, while the second one, `source_ad.cxx`, accesses the AD card. It is different from `source.cxx` as it needs to perform also control settings (sampling frequency and frame buffer allocation) on the card.

Despite the apparent complexity of the integration effort, the reality turned out that this approach was fast and efficient. Once the device driver and the source actor were developed, the code of the entire application (30 Klines C++, including transmitter model, channel model, and receiver ASIC model) could be ported without additional modifications. One possible alternative approach would be the use of a prototyping development environment such as LabViews. In that case, the device driver development would have been saved, but on the other hand, the complete application would need porting.

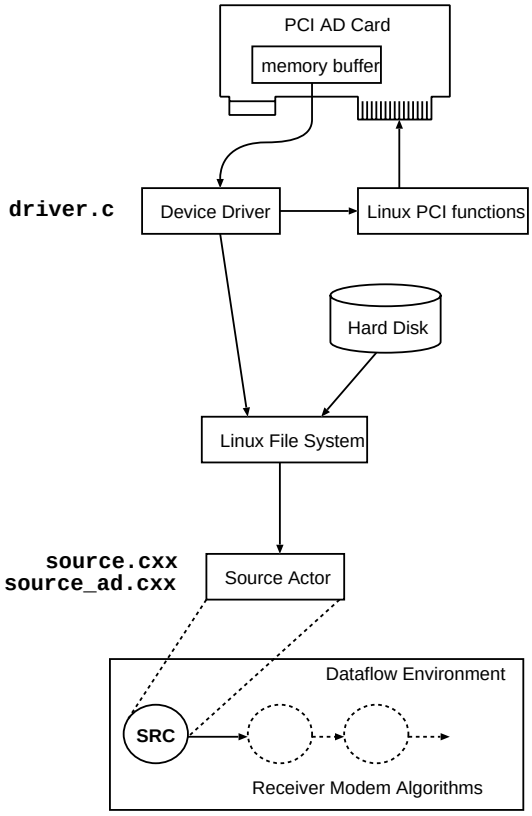

Figure 6. Aqcuisition Hardware Integration

Having this environment in place in our development environment, doing prototype experiments with an ASIC model is a matter of minutes: we have to replace the `source` and `sink` actors in the data-flow environment

with the appropriate acquisition hardware actors, and recompile the C++ ASIC description on the prototyping PCs.

5. Conclusions

Finally, the contributions of this work are summarized. We proposed an integrated approach to the development of complex modem ASIC algorithms and their prototyping. The efforts needed for this integration are low because the prototyping aspect is integrated into the end-to-end model, and allows the ASIC model to remain almost unmodified. Although it does not offer real-time operation, the approach is well suited for functional verification in the real environment.

Acknowledgements

This work was done under the Flemish Impulse Program for Information Technology (IT-ISIS). The cable modem was developed as a joint effort of Siemens-Atea (Herentals), IMEC and the Katholieke Universiteit Leuven. Mark Engels is a senior research assistant of the NFWO.

References

[1] http://www.siemens.be/atea/products_services/rd_technology/rd_technology.htm.

[2] M. Courtoy. Project spinnaker: A new generation of rapid prototyping system. In *Fifth Intnl Workshop on Rapid System Prototyping*, pages 141–144, 1994.

[3] M. Dahl, J. Babb, R. Tessier, S. Hanono, D. Hoki, and A. Agarwal. Emulation of the sparcle microprocessor with the mit virtual wires emulation system. In *IEEE Workshop on FPGAs for Semicustom Computing Machines*, pages 14–22, 1994.

[4] R. Lauwereins, M. Engels, M. Ade, and J. Peperstraete. Grape-ii: A system-level prototyping environment for dsp applications. *IEEE Computer*, pages 35–43, February 1995.

[5] J. Pino, S. Ha, E. Lee, and J. Buck. Software synthesis for dsp using ptolemy. *Journal on VLSI Signal Processing*, 1993.

[6] J. Yoo, E. Brunvand, and K. Smith. Automatic rapid prototyping of semi-custom vlsi circuits using actel fpgas. In *Fifth Intnl Great Lakes Symposium on VLSI*, pages 148–151, 1995.