A 10 Mbit/s Upstream Cable Modem with Automatic Equalization

Patrick Schaumont Radim Cmar Serge Vernalde Marc Engels
IMEC vzw, Kapeldreef 75, B-3001 Leuven Belgium

Abstract

A fully digital QAM16 burst receiver ASIC is presented. The BO4 receiver demodulates at 10 Mbit/s and uses an advanced signal processing architecture that performs per-burst automatic equalization. It is a critical building block in a broadband access system for HFC networks. The chip was designed using a C++ based flow and is implemented as a 80 Kgate 0.7u CMOS standard cell design.

1 Introduction

The widespread use of Internet is opening a pathway to emerging multimedia consumer networks and applications. These require a broadband data communications link to be established in the access network that connects the consumer to a core service network. The hybrid fiber-coax (HFC) access network that is currently in use for cable TV, is considered as an attractive candidate [4]. We have developed a chip that is embedded in an HFC head-end and that demodulates data transmitted from the consumer set-top. This chip is a fully digital burst receiver, characterized as shown in table 1.

The chip design will be described as follows. In section 2, we present the system level architecture and design choices. Next, section 3 will elaborate on the design flow and C++ modeling that was applied, including VHDL code generation and synthesis. Section 4 highlights the verification strategy. We present the obtained prototypes and measurement results in section 5, and conclude in section 6.

Functional Properties		
Modulation	QAM16/QPSK	
Multi-access	TDMA	
Preamble Length	17	symbols
Min Burst Spacing	4	symbols
Payload Length	1-1024	bytes
Symbol Rate	2.56	Msym/s
Bit Rate	10.24/5.12	Mbyte/s
Input Sample Rate	10.24	MHz
Input Bandwidth	3.3	MHz
AGC range	10	dB
RRC Shaping	adaptive	
Equalization	per-burst, fully autonomous	
Performance at C/N = 22.5 dB		
Uncoded BER	5^{-5}	
Coded BER	10^{-10}	
Standard Compliance		
	MCNS/DOCSIS	
	IEEE802.14	
	DAVIC/DVB	
	ITU-T J.112	
Interfaces		
Error Correction	Reed Solomon	
Programming	I^2C	
Data Connection	UTOPIA	

Table 1: Chip Properties

2 System Level Architecture

2.1 System context

The system level architecture in which our receiver is embedded is shown in figure 1, which illustrates HFC upstream communications for a concrete application scenario. A user connects a PC to the access network using a cable modem. Among other tasks, this device modulates a digital message from the PC into a QAM16 signal burst. Burst modulation enables a TDMA multiaccess scheme in which multiple users can be connected to the same head-end simultaneously. The burst is characterized by a preamble that synchronizes the head-end receiver, a payload that contains the actual data message, and a carrier frequency.

At the head-end side, the received signal is passed through an analog front-end `AFE`. The front-end converts the received carrier to a low, fixed carrier frequency. Next, the signal is digitized by an analog-to-digital converter `AD`, and digitally demodulated in the receiver chip `BO4`. The demodulated message is then passed on to a cell transport system `CTS` connected to the core network. Both the receiver chip and the analog front-end are under control of a medium access control component `MAC`, that selects the burst transmission frequency and time slot.

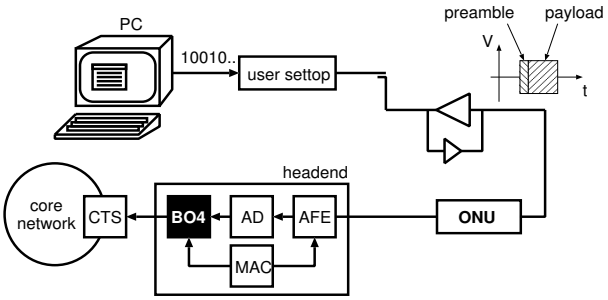

Figure 1: System Level Architecture

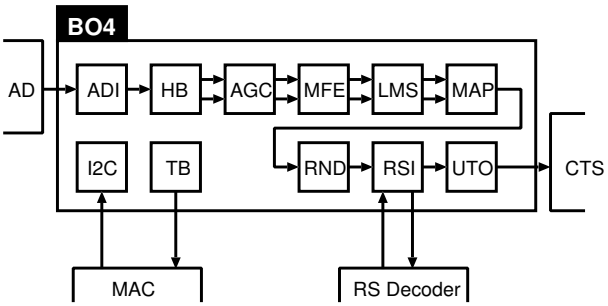

Figure 2: Beaufort RX Architecture

2.2 Digital Receiver Architecture

The patented architecture of the BO4 receiver chip is shown in figure 2. The demodulation of the QAM16 burst signal is done by a chain of loosely coupled signal processors. The burst signal enters the chip through an A/D converter interface `ADI`. `HB` down-converts this signal to baseband. Next, `AGC` normalizes the signal power level, and `MFE` constructs a matched filter for QAM16 detection. An adaptive equalizer `LMS` removes remaining intersymbol interference (ISI). Subsequently, `MAP` converts the QAM16 symbols into a byte sequence. This byte sequence passes through an interface `RSI` for off-chip channel decoding, and finally enters the cell transport system through a UTOPIA bus interface `UTO`. Chip programming is done with an I^2C standard interface `I2C`, while various internal signals are observable through a testbus interface `TB`.

2.3 Block Architecture

The architecture of the individual blocks was devised according to a standard architecture template. Each of the blocks consists of one or more interconnected FSMD, as shown in figure 3. Each FSMD is made up of a finite state machine `FSM` and a bit-parallel synchronous datapath `DP`. `FSM` and `DP` exchange instructions and status signals.

The local control FSM exchanges two types of control signals with the rest of the system: globally generated control signals, and signals that are send along with the flow of data,

Global control signals include reset and clock, as well as rate control signals. The rate control signals are used to synchronize the distributed block schedules to a common reference, and therefore implement the static dataflow part of the chip. The rate control signal for each block varies with the data introduction interval (DII [3], the number of clock cycles per data sample). Inside of BO4, the DII gradually

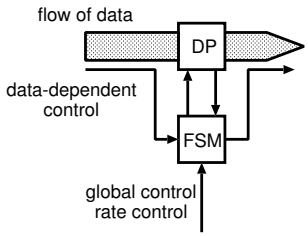

Figure 3: FSMD model

decreases along the signal processing chain, from $f_{clock}/4$ at the ADI interface downto $f_{clock}/32$ at the UTO interface.

Many of the blocks also process signal samples conditionally. For instance, the MFE block will only operate after the AGC block has detected a burst start. Another example is the interaction of the I^2C interface with blocks in the main signal processing chain. For this conditional data processing, control signals are send along with the data to indicate signal sample presence.

Finally, datapath register update at various rates is done with a synchronous strategy. The update control signals are evaluated in the local FSM as a combination of local sequencing, global rate-control, and global data-dependent control.

3 Design Flow

The design flow of the BO4 chip is shown in figure 4. The flow contains three major parts: a system level design part, a hardware synthesis part and a hardware verification part. This section is concerned with the system level design and hardware synthesis issues, while section 4 focuses on verification.

3.1 System Design

The goal of the system design phase is to construct a functional RT-level model of the BO4 chip. For verification and test purposes, an end-to-end model is however required. This end-to-end model includes, besides the BO4 receiver model, a transmitter model to generate test bursts and a channel model to distort the test bursts according to the expected transmission impairments. The impairments include those of coax, distribution amplifiers and analog front-ends.

We use C++ as our primary system design environment, since it allows to mix the high level environment model with the detailed architecture model of the target receiver. We use a design environment [5] that supports simulation of high level dataflow as well as cycle true architecture models. In addition, it has an elaborate code generation backend that allows a smooth transition to circuit synthesis and verification.

Initially, a floating point data flow model of the complete system is constructed (transmitter, channel model, receiver). Next, the BO4 receiver is refined to a cycle true architecture model. This is done by scheduling the operations of high level descriptions to clock cycles. Since the cycle budgets of the most complex blocks (`MFE` and `LMS` equalizers) have only 8 cycles, scheduling can be done by hand without much trouble. In addition, bringing dataflow to hardware also requires the mapping of the dataflow computational model (firing rules) to an implementation. For this purpose, we make use of rate-control and data-dependent control signals (as explained in figure 3).

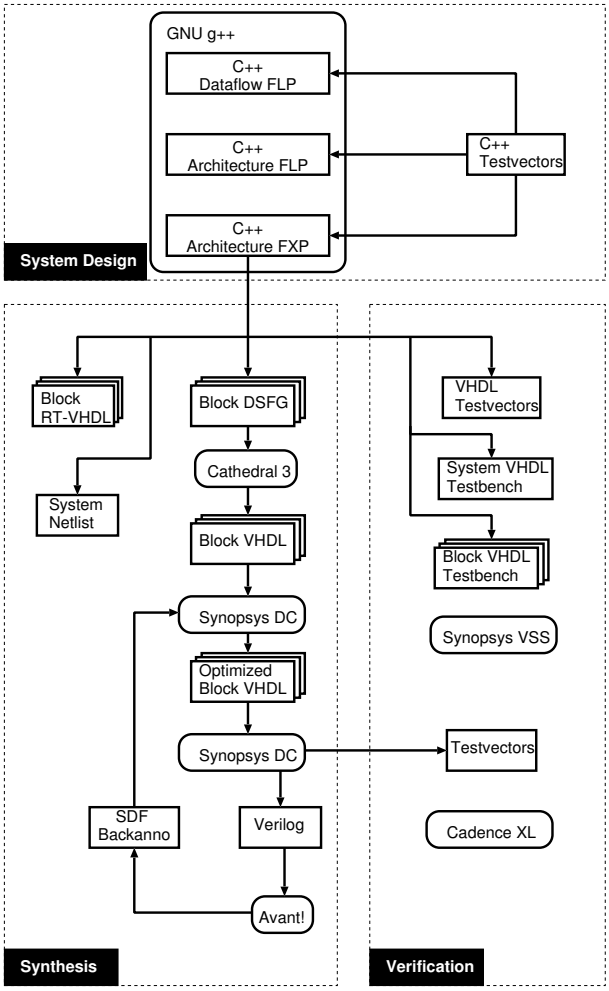

Figure 4: Design Flow

After the architecture has been obtained, the chip signal wordlengths are decided in order to yield a cycle true, bit-true architecture model. Fixed point refinement is done by means of simulation. A reception quality metric, constellation purity, is first determined using only quantization at the A/D side (10 bit). Next, the other wordlengths are decided such as to prevent overflow and to maintain the reception quality metric.

After these steps, the C++ model is a bit-true. clock-cycle true representation of the architecture. Now, a code generator creates the input for subsequent hardware synthesis and verification.

- For each block (FSMD) of the receiver, an synthesizable RT-VHDL file is created. During the BO4 design, it is primarily used for verification.
- For the overal BO4 chip, a system netlist is generated to connect the various FSMD blocks.
- For each FSMD, code is generated for the Cathedral 3 [1] synthesis tools, which contain advanced word-level operation-sharing algorithms and allow to generate technology-mapped FSMD descriptions.
- Finally, the C++ testvectors are also translated into FSMD-level and system-level testbench vectors and drivers. These allow to perform verification of subsequent synthesis results.

Specification		
C++ Dataflow	922	lines
C++ Architecture	4426	lines
C++ System Testbench	11014	lines
RT VHDL	21798	lines
Gate Level VHDL	154952	lines
Architecture: Single-clock Synchronous		
FSMD	26	
Clock Frequency	20.48	MHz
Gate Count	83130	
Standard Cell Count	38768	
IO Pins	95	
Test Coverage	99.27	%
Scan (Full-scan)	6	chains
Test Vectors	472	
Layout and Technology		
Technology	0.7	μ
VDD	5	V
Core	80.7	mm^2
Chip	113.9	mm^2
Active Area	37.4	mm^2
Metal Layers	3	
Routing Factor	2.15	

Table 2: Chip Metrics

3.2 Synthesis

Circuit synthesis is a fully automated process. Using elaborate scripting, a verified gate-level system netlist is obtained out of the generated code within 36 hours. The synthesis tools are run on a HP-9000 series workstation with 2 gigabyte of internal memory.

The synthesis is a multi-stage process taking the following steps.

- Each FSMD is processed by Cathedral-3 to yield an operator-level technology-mapped netlist.
- Next, it is processed by Synopsys DC to perform logic optimization, and to insert scan chains for production test. We use worst-case commercial operating conditions. For timing verification, we use a standard wire-load model for the first iteration, and subsequently a capacitance load file produced by the layout backend.
- A new run of Synopsys DC is done to convert the resulting blocks to Verilog, and to produce test vectors by ATPG.
- The resulting Verilog netlist is processed by the Avant! layout backend, producing a capacitance load file (SDF). This load file is used to verify the logic optimization of Synopsys DC.

Some chip metrics that are obtained out of the synthesis process are collected in table 2.

4 Verification Strategy

During synthesis, simulation-based verification is used extensively to track the correctness of the synthesis results. All VHDL-level simulations are done using the generated testbenches with the Synopsys VSS simulator at block-level and system level. The final Verilog netlist is checked using generated production test vectors with the Cadence Verilog-XL simulator.

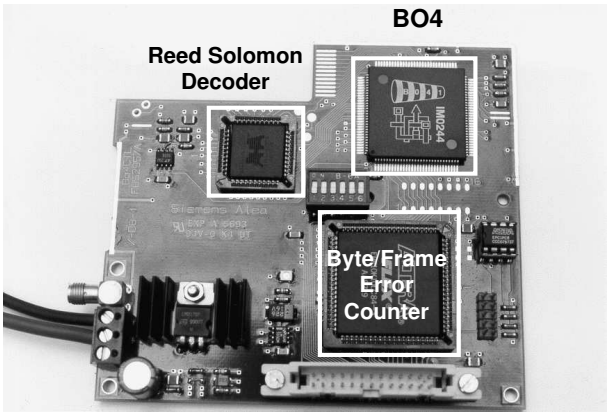

Figure 5: Receiver Board with BO4

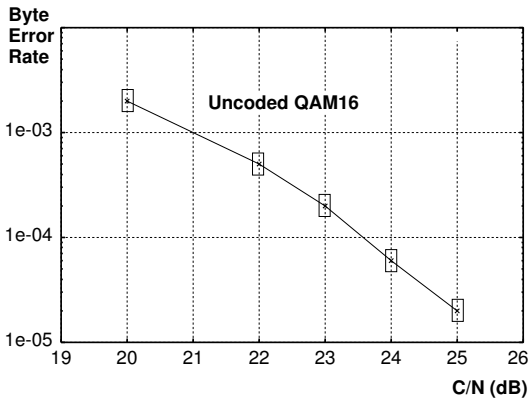

Figure 6: Measured Receiver System Performance

Verification is done by C++ simulation during the system design phase and by HDL simulation during the synthesis phase. There are 7 verification levels that correspond to the 7 description levels of the design. Three of them are in C++ (dataflow floating-point, cycle-true floating-point and cycle-true fixed point). The remainder four are at VHDL (RT-VHDL, Cathedral 3 and Synopsys-DC VHDL outputs) and Verilog (final netlist) level.

The design of testbenches is done in C++, since corresponding HDL testbenches are obtained by code generation. The test simulations can be categorized in three areas: Performance tests, functional tests, and equivalence tests.

The performance tests are used to check the initial performance of BO4 in terms of bit-error rate and constellation purity. Test scenarios include varying levels of channel noise, phase distortion, carrier frequency deviation, amplitude slope distortion, gain variation and burst spacing. These tests ensure that the initial algorithmic model has the desired performance.

The functional tests check the correct operation of BO4 within one verification level. Typical tests include for instance the reception of a known data sequence. The goal of these tests is to perform a simulation with maximal coverage of the design description. For this purpose, our C++ design environment allows to obtain simulation coverage measures. After a C++-level architecture simulation, the FSMD descriptions are interrogated to return the number of times an arbitrary FSM transition has triggered. In addition, statistics are collected on the signals of the datapath description regarding the number of reads, writes, and signal ranges. This way, a test suite is constructed that exercises a maximal part of the description.

Equivalence tests compare the operation of one level to the next. They are applied at either floating-point level or else fixed-point level. Equivalence tests do a one-to-one comparison of values on the system interconnect at corresponding time-points.

5 Prototyping

Figure 5 shows a prototype PCB that uses the BO4 chip. This board contains also an reed solomon decoder device, and a real-time byte/frame error counting FPGA used for verification purposes. A real-time setup [2] based on this board allows to characterize the chip in detail.

An example performance measurement done at a test site, consisting of 2 sections of taps with return amplifiers and 7 trunk amplifiers, is shown in figure 6. The measurement shows the byte error performance as a function of channel noise for the overall system including a commercial transmitter, analog front-ends, and the BO4 receiver.

6 Conclusions

The BO4 chip, which is an upstream HFC receiver chip, was presented. This chip uses state-of-the-art signal processing to achieve QAM16 communication with good performance. A critical enabler for this is the C++-based design flow that integrates system design and circuit design. This resulted in a short design time and first-time-right silicon.

7 Acknowledgements

This work was carried under the Flemish Impulse Program for Information Technology (IT-BAN). The results reported in this paper is would not have been possible without the continuous effort of many other persons in this collaboration. Specifically, we acknowledge the contributions of Willy Trog and Karl De Meyer from Siemens Atea, Herentals, Belgium, during system design. Throughout the project, there was also close and continuing cooperation with SISTA, Leuven University, Belgium (Algorithms), ACCA, Leuven University, Belgium (DSP Rapid Prototyping), and INTEC, Ghent, Belgium (Hardware Prototyping).

References

[1] W. Geurts, F. Catthoor, S. Vernalde, and H. Deman. *Accelerator Data-Path Synthesis for High-Throughput Signal Processing Applications*. Kluwer Publishing, 1997.

[2] Siemens Atea R&D Technology Homepage. http://www.siemens.be/atea/products_services/rd_technology/rd_frames.htm.

[3] H. S. Jun and S. Y. Hwang. Design of a pipelined datapath synthesis system for digital signal processing. *IEEE Trans. VLSI Syst.*, 2(3):292–303, September 1994.

[4] W. Pugh and G. Boyer. Broadband access: Comparing alternatives. *IEEE Communications Magazine*, pages 34 – 46, August 1995.

[5] P. Schaumont, S. Vernalde, L. Rijnders, M. Engels, and I. Bolsens. A programming environment for the design of complex high speed asics. In *Proceedings 35th Design Automation Conference*, pages 315 – 320, San Francisco, CA, 1998.