HARDWARE-SOFTWARE CODESIGN OF A 14.4MBIT - 64 STATE - VITERBI DECODER FOR AN APPLICATION-SPECIFIC DIGITAL SIGNAL PROCESSOR

Michael Hosemann, Rene Habendorf, and Gerhard P. Fettweis

Vodafone Chair Mobile Communications Systems
Dresden University of Technology
01062 Dresden, Germany

{hosemann, habendor, fettweis}@ifn.et.tu-dresden.de

ABSTRACT

Viterbi Decoders are employed frequently in wireless radio systems. They often require high computational efforts which can only be handled by dedicated application specific integrated circuits *(ASIC)*s. Because of their flexibility and speed of development, DSP-based software solutions are desirable, however. Currently available DSPs are not able to provide enough computational power to perform the Viterbi decoder for systems such as digital video broadcasting *(DVB)* where a 64-state Viterbi decoder needs to be performed for 4 .. 30 MBit/s together with the other required receiver algorithms. Hence, we strive to provide means of increasing DSPs computational powers. One way for increasing computational power is to provide multiple data paths, operating in parallel, in a processor.

Two different methods of parallel computation of a Viterbi decoder are presented and their requirements for the DSP architecture are analyzed. These methods are not only applicable for DVB the Viterbi decoder in DVB systems but for all terminated convolutional codes. Following, we are deriving a data path design for a highly parallel DSP. We introduce special instructions which not only speed up the computation of Viterbi decoders but are also beneficial for computing a Fast Fourier Transform. This data path design can calculate an add-compare-select butterfly in two cycles and thus allows the DSP to perform the computation of the Viterbi decoder for the current German variant of DVB-T (14.4Mbit/s) at a modest 70 MIPS. This leaves enough computational power to also perform the other receiver algorithms at a targeted clock rate of 200MHz.

1. INTRODUCTION

Viterbi Decoders are a popular technique for decoding convolutional codes which are employed frequently in many wireless radio systems. One such system is digital video broadcasting *DVB* currently introduced in Europe employing a 64-state Viterbi decoder at data rates of 4..30 MBit/s,.

At high data rates and numbers of states Viterbi decoders require high computational efforts which often can only be handled by dedicated application specific integrated circuits *(ASICs)*. However, because of their flexibility and speed of development DSP-based software solutions are desirable. Currently available DSPs are, however, not able to provide enough computational power to perform the Viterbi decoder for DVB together with the other algorithms required for a complete DVB-T receiver implementation. Consequently, we are seeking ways of designing DSPs which are able to perform such computationally demanding algorithms.

A feasible way for increasing the computational resources is exploitation of parallelism. Particularly, if single-instruction multiple-data *(SIMD)* control schemes can be used, parallelism allows for fast and efficient implementations of computationally demanding algorithms.

However, this is only possible if the algorithm to be performed contains enough independent operations which can be executed in parallel. Hence, we are investigating ways of performing the Viterbi decoder of a DVB receiver by means of parallel computing. First, two different ways of parallel computation are presented. Following, we are deriving a data path design for a highly parallel DSP which can perform these computations at a modest 70 MIPS.

This paper is organized as follows: In the remainder of this introduction we will introduce the relevant system properties of the DVB system. The basics of the Viterbi decoder and the key features of our scalable, parallel DSP architecture are explained in Sections 2 and 3, respectively. Section 4 presents methods of parallel implementations of the Viterbi decoder. In Section 5 we derive the data path implementation for our DSP and present implementation results.

This work was sponsored in part by Deutsche Forschungsgemeinschaft (DFG) within SFB358-A6

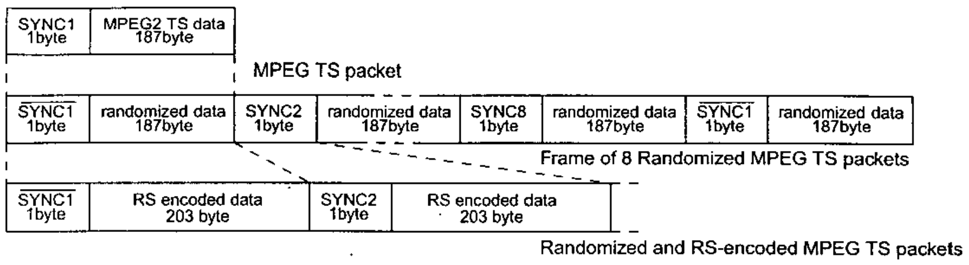


Fig. 2. Frame Structure of Transmitted Data

1.1. Digital Video Broadcasting

Digital Video Broadcasting *DVB* is a television broadcasting system planned to replace the current analog broadcasting schemes [3]. There are three different variants of the system, satellite (DVB-S), cable (DVB-C) and terrestrial (DVB-T), with terrestrial broadcasting being the computationally most complex variant. Additionally, DVB-T is also considered as a supplement to UMTS for broadcasting data which are of interest to a large number of users. This makes it particularly interesting for implementation in a software radio solution since a mobile terminal would benefit a lot from a software implementation running on a DSP. Commercially available solutions so far employ dedicated ASICs ([4] and others) or a large number of DSP chips and hardware accelerators [5]. Figure 1 shows a block diagram of a DVB-T transmission system with the highlighted Viterbi decoder. Together with the OFDM synchronization and demodulation the Viterbi decoder accounts for about 90% of the computational requirements. Hence, an efficient and fast implementation is crucial for any signal processing device performing DVB-T reception. The transmitted data are organized in frames as depicted in Figure 2. Each frame of 187 Byte of MPEG-encoded is marked with a preceding synchronization byte. During energy dispersal 8 consecutive of these frames are treated as one super-frame where the first synchronization byte is reversed bitwise. During the following Reed-Solomon encoding parity bytes are appended to the frames but the synchronization bytes are not changed. The subsequent convolutional outer interleaver does not change the spacing of the synchronization bytes of 204 bytes either. Hence, there is a known symbol every 204 symbols which essentially means a termination of the convolutional code. Note, however, that only an OFDM frame of 68 OFDM Symbols always begins with a reversed synchronization byte. This can be exploited for the time-parallel implementation of the Viterbi decoder as explained in Section 4.2.

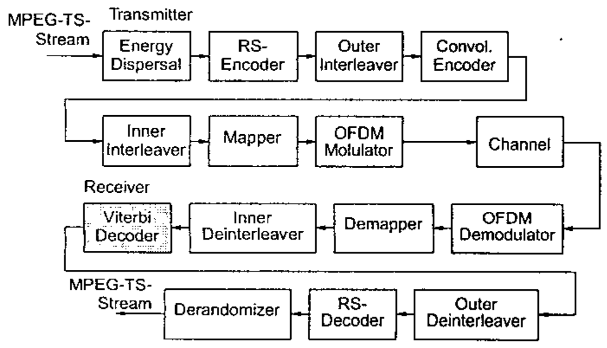


Fig. 1. Block Diagram of DVB-T Transmission System

2. VITERBI DECODERS

In general, a convolutional code $C(n, k, [m])$ can be described by a linear finite-state shift register with k-dimensional input and m stages [1]. The n algebraic function generators compute the n-dimensional output. The code rate is defined as $R_C = k/n$, the constraint length of a convolutional code is $L_C = m + 1$.

The Viterbi Algorithm is a method to decode convolutional codes and is frequently expressed in terms of a trellis diagram as depicted in Figure 3. This two dimensional graph is described in vertical direction by the N states of the finite state machine and in horizontal direction by discrete time instants ν. The states of two consecutive time instances are connected by branches representing the state transitions. The main part of the Viterbi decoder is the add-compare-select *(ACS)* recursion which is described following: For all branches in one time interval (ν, $\nu + 1$) the branch metrics $\lambda^{\nu}_{j,i}$ are computed and accumulated to obtain path metrics for each path through the trellis (add-part). By selecting the "most likeliest" path, i.e. the path with the maximum path metric $\gamma_j^{\nu+1}$ for every state j=0..N-1, the number of paths in the trellis is constantly N.

For common convolutional codes with one dimensional input each state in the trellis has two incoming and two out-

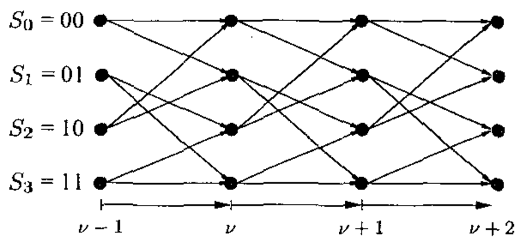
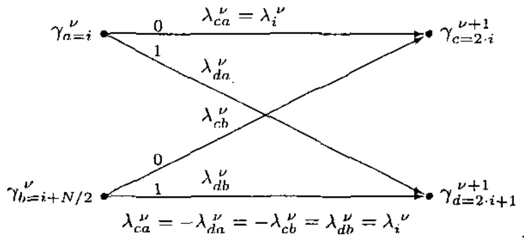


Fig. 3. Trellis Diagram of $C(*, 1, [2])$ Convolutional Code

going branches. Hence, the ACS recursion can be divided into $N/2$ ACS butterflies depicted in Figure 4.



$\lambda_{ca}^{\nu} = \lambda_i^{\nu}$
$\gamma_{a=i}^{\nu}$
0
1
λ_{da}^{ν}
λ_{cb}^{ν}
0
1
λ_{db}^{ν}
$\gamma_{c=2\cdot i}^{\nu+1}$
$\gamma_{b=i+N/2}^{\nu}$
$\gamma_{d=2\cdot i+1}^{\nu+1}$
$\lambda_{ca}^{\nu} = -\lambda_{da}^{\nu} = -\lambda_{cb}^{\nu} = \lambda_{db}^{\nu} = \lambda_i^{\nu}$

Fig. 4. ACS butterfly

The path metrics $\gamma_{c,d}^{\nu+1}$ are computed by

$$\gamma_c^{\nu+1} = max(\gamma_a^{\nu} + \lambda_i^{\nu}; \gamma_b^{\nu} - \lambda_i^{\nu}) \quad (1)$$
$$\gamma_d^{\nu+1} = max(\gamma_a^{\nu} - \lambda_i^{\nu}; \gamma_b^{\nu} + \lambda_i^{\nu}) \ . \quad (2)$$

The ACS recursion consists of $2N$ add and N compare and select operations. For each decoded bit one ACS recursion has to be computed, therefore it is the bottleneck of the Viterbi Algorithm. Note, that also the information about the survivor of the selection has to be stored in a path memory.

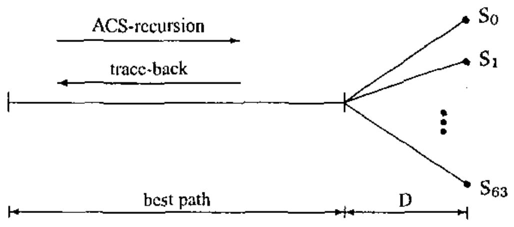


Fig. 5. trace-back and survivor depth [2]

After computing a sufficient number of time instances all N paths are traced back until they merge into one path. The trellis depth at which all paths merge with sufficiently high probability is referred to as the survivor depth D and describes the minimum latency of the Viterbi Algorithm. With every following recursive step one bit is decoded depending on the survivor information in the path memory.

The DVB-T standard ([3]) defines a $C(2, 1, [6])$ convolutional code with N=64 states. The available punctured rates range from 2/3, which is used in Germany, to 7/8. According to simulation results for the DVB-T convolutional code with a punctured rate of 2/3 a survivor depth of about 100 is recommended.

3. SCALABLE DSP ARCHITECTURE

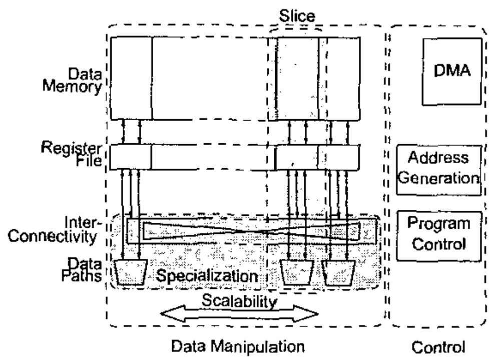


Fig. 6. Overall Architecture of the Platform-Based DSP

We are designing DSP architectures following the concepts presented in [6]. The architecture will be derived from a platform by scaling the number of slices and tailoring the functionality of these slices. Additionally, the communication network between these slices has to be considered since it can require a large amount of chip space and introduce long delays. Figure 6 shows the overall architecture of our platform based DSP. The data manipulation part consists of a scalable number of slices, each containing data memory, a register file, a part of the interconnectivity unit *(ICU)* and a data path. The ICU and data path are tailored to the functionality required by the target algorithms. Such functionality could be a Viterbi-butterfly-tailored network in the ICU, special arithmetic like Galois-field in the ALU or the special capabilities for the Viterbi decoder described in this paper.

The control part performs program control, address generation and direct memory access *(DMA)*. All Slices are controlled by just one program control unit in SIMD fashion. This means that while adjusting the number of slices to fulfill the computational requirement control overhead remains constant. However, this also implies limitations in the parallelism that can be exploited in the target application. In Section 5 we will show how some small additions can allow for more flexibility and improved performance despite the limitations of SIMD.

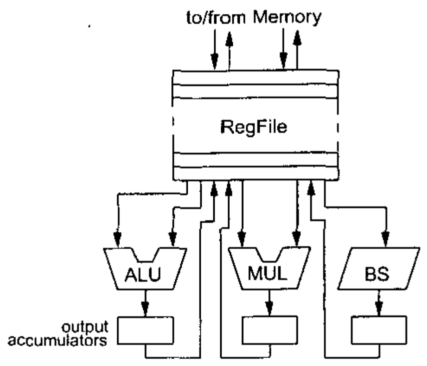


Fig. 7. Architecture of one Slice's Data Path

The basic architecture of the data path of one slice is depicted in Figure 7. The processor will feature mainly 16 bit integer operation. A register file (RegFile) stores data while the arithmetic-logic unit *(ALU)*, multiplier *(MUL)*, and barrel shifter *(BS)* are performing the actual operations. Each of these units features an accumulator register of a width matching the maximum output of the respective unit. These accumulators can serve as input registers for all data manipulation units allowing for consecutive operations on data wider than the registers of the register file. For clarity these connections were omitted from Figure 7.

The multiplier is not required for the Viterbi decoder but necessary for a DSP which will perform other applications, too. In Section 5 we will derive special features for our data path in order to enhance the performance for Viterbi decoders.

4. PARALLEL IMPLEMENTATION OF THE VITERBI DECODER

4.1. Parallization Over States

Since there are no data dependencies between the ACS butterflies in one time interval $(\nu, \nu+1)$, the ACS recursion can be split over the different data paths. Every slice computes a subset of the 32 ACS butterflies.

However, the resulting path metrics of one ACS butterfly are needed in different ACS butterflies in the next ACS recursion. Therefore, a communication network is required to rearrange the path metrics over the slices. Figure 8 shows a possible network for 16 slices and N=64 states.

With 16 bit per path and up to 8 paths in parallel in horizontal direction this network requires a large amount of chip space and power. Realizing the network in several stages would introduce long delays. Hence, it is desirable to find another way of speeding up the computation.

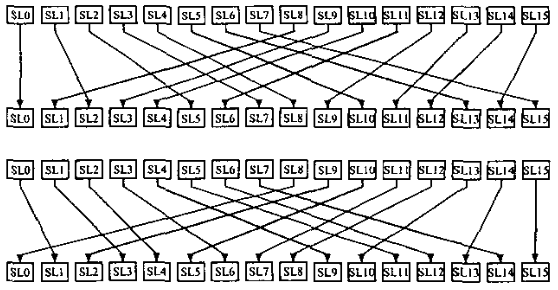


Fig. 8. Required Inter-Slice Communication Network

4.2. Parallization Over Time

In [2] it was shown that a convolutionally encoded data stream can be divided into consecutive blocks which can be decoded independently in parallel. For non-terminated convolutional codes this results in additional computational efforts to find the best path in each block.

As explained in Section 1.1 the DVB-T convolutional code is terminated over blocks of 204 bytes, the Reed-Solomon Packets *(RSP)*. By building up the whole trellis for one RSP the trace-back can start from the state given by the termination sequence. Therefore, no tracing back over the survivor depth D before decoding is necessary. Hence, different blocks can be decoded separately on different slices. However, to use the termination the synchronization sequence must be arranged at the end of each block. Since the synchronization bytes are the first bytes of each RSP, the blocks sent to the Viterbi decoder are shifted RSPs, further denoted as SRSP. .

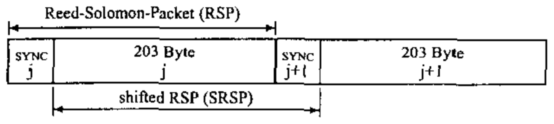


Fig. 9. Reed-Solomon-Packet and shifted RSP

For one SRSP 1632 $(204 \cdot 8)$ ACS recursions, each including 32 ACS butterflies, have to be computed before the best path can be traced back. For each state in each time instance the survivor is coded with one bit. The survivors of one ACS recursion are stored in four 16 bit data words, further denoted as trellis word. The path memory for a complete SRSP requires

$$1632 \text{ ACS-rec} \cdot 64 \frac{\text{bit}}{\text{ACS-rec}} = 104.448 \text{ bit.} \quad (3)$$

For 16 slices a path memory of 1,67 Mbit is needed.

Assuming four bit soft decision input for the Viterbi decoder, after 1632 ACS-recursions the path metrics range from

$$-16 \cdot 1632 = -26112 \leq \gamma^{\nu} \leq 14 \cdot 1632 = 22848 \ . \quad (4)$$

Since the path metrics are stored in 16 bit data words, no normalization is required.

After building up the trellis the best path is traced back, therefor the recursive path information of the preferred state, the survivor bit, is evaluated. The preferred state is given by the termination sequence. Depending on the survivor bit the previous state $S^{\nu-1}$ in the best path is computed and the respective bit of the SRSP is decoded. $S^{\nu-1}$ indicates the position of the survivor bit in the next trellis word.

To trace back one SRSP 1632 recursive steps (one for each bit to be decoded) through the trellis have to be computed. Each step requires the evaluation of the survivor bit from the trellis word, the computation of the next state and the decoding of the information bit according to the respective transition. Assuming at least five cycles for one step a sequential computation demands

$$14.4\text{Mbit/s} \cdot 5\ \text{instructions/bit} = 72\ \text{MIPS} \tag{5}$$

to meet the DVB-T requirements as explained in Section 5. To minimize these computational efforts a parallel realization of the trace-back is desirable. Since the best paths differ between the slices, different data words of the respective 64 bit trellis word have to be evaluated on each slice. Hence, a realization with SIMD instructions requires a conditional execution of operations where an instruction is executed only on slices bearing a certain condition code generated by a previous instruction.

Furthermore, the data path has to be optimized for the ACS butterfly to achieve the required throughput as explained in Section 5.

Of course the presented method introduces an additional latency of 16 RSPs equaling approximately 2ms. This seems still acceptable for a broadcasting scheme.

5. DATA PATH DESIGN

For our ongoing DSP design we are aiming at a clock frequency of 200 MHz. Hence, considering the computational efforts introduced by the OFDM synchronization and demodulation as other algorithms with high computational requirements the Viterbi decoder should be computed with less then 100 MIPS.

The DVB-T data rates range from 4 to 32 Mbit/s [3], in Germany 13.27 Mbit/s are used. According to a punctured rate of 2/3 the data rate after the Viterbi decoder is 14.4 Mbit/s. We are using this rate as a goal for our design for which no custom layout will be feasible. By using a more sophisticated semiconductor technology it likely would be possible to achieve clock rates and computational performances allowing for all modes of operation.

Design for ACS recursion As already mentioned, our algorithm computes one SRSP with a length of $8 \cdot 204 = 1632$ bit per slice. For each ACS butterfly (ACS-B) four add and two compare and select operations have to be done.

$$\gamma_{1,2} = \gamma_a^{\nu} \pm \lambda^{\nu} \quad \text{and} \quad \gamma_{3,4} = \gamma_b^{\nu} \pm \lambda^{\nu} \tag{6}$$

$$\gamma_c^{\nu+1} = max\,(\gamma_1; \gamma_4) \quad \text{and} \quad \gamma_d^{\nu+1} = max\,(\gamma_2; \gamma_3) \tag{7}$$

With common instructions an ACS-B will cost at least 6 cycles. This results in a computational effort of

$$\frac{14.4 Mbit/s}{16 \cdot 1632\, bit} \cdot 1632 \cdot 32\ \text{ACS-B} \cdot \frac{6\ \text{instructions}}{\text{ACS-B}} = 172\ \text{MIPS}\,. \tag{8}$$

To meet the requirements of less than 100 MIPS for the complete Viterbi algorithm the ACS butterfly has to be decreased to two cycles. In the first cycle the ALU computes the four add operations with a special double-add-sub instruction (DADDSUB). Therefore, the ALU requires three input operands in one cycle. Since all metrics are coded with 16 bit, the first 32 bit of each ALU port can be divided into high and low word to store two metrics as shown in Figure 10.

It shall be noted that this instruction does not only greatly accelerate the Viterbi ACS butterfly but can also be used for computing a Fast Fourier Transform (FFT), where similar calculation patterns can be found.

In the second cycle the ALU computes the two compare and select operations with a special compare instruction (DMAX). This is performed by calculating the difference of two inputs and selecting the larger one by using the sign bit of the resulting difference. Again this is not a purely Viterbi-dedicated instruction but it can be used for other applications too.

The two maximum path metrics are passed to the ALU output registers and for each decision a flag is set to code the respective survivor. These flags have to be arranged into 16 bit data words, therefor an additional shift operation (SHIFTFLAG) was added to the barrel shifter's instruction set. The operand is shifted by two and filled with both flags. After 8 ACS-butterflies this data word contains survivor information for 16 paths and it is stored in the path memory afterwards.
With these changes the ACS recursions can be computed with 67 MIPS.

Design for Trace Back As explained in Section 4.2, conditional execution of operations is required to trace back the 16 SRSPs in parallel. Hence, we propose conditional instructions which execute only on slices where a condition flag is set. This condition flag can be any of the common zero, sign, carry, or overflow flags which are generated by the ALU of each slice. For if-else statements containing multiple ALU instructions per branch, the flag generation

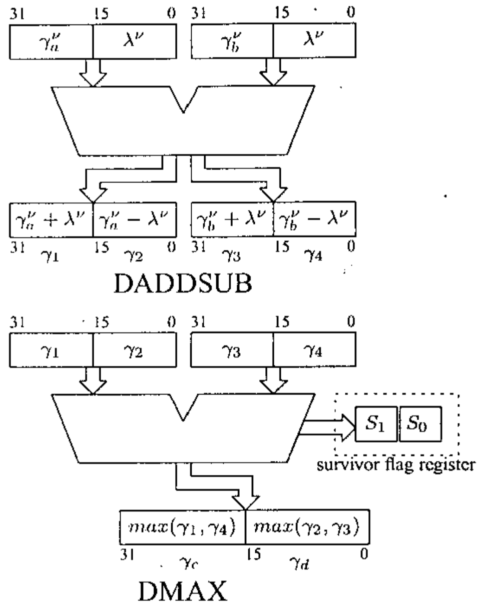


Fig. 10. Extended ALU with special Instructions

must be conditional too, specified in the instruction word. The structure of an instruction looks like:

INST <flag generation> <condition flag> OP1 OP2 OP3.

Employing all these features, a simple SIMD if-then-else structure can then be realized in three cycles as can be seen in the example in Table 1. Here, the execution depends on the zero flag (ZF).

Conventional C-like	*Conditional Assembly*
if $A < B$	SUB A B
then LOAD REG1	(ZF) LOAD REG1
else LOAD REG2	(!ZF) LOAD REG2

Table 1. Example of Conditional Execution

For the trace back of the 16 SRSPs in parallel 6 MIPS are required. This results in a computational effort of 73 MIPS for the Viterbi Algorithm which meets our demands. Table 2 summarizes the memory requirements for the implementation on 16 slices.

6. CONCLUSIONS

We presented ways of computing the Viterbi decoder required for DVB-T in a parallel manner. However, the pre-

soft input values:	$(1632 \cdot 2 \cdot 4) \cdot 16$	=	208.896 Bit
path metrics:	$(64 \cdot 16 \cdot 2) \cdot 16$	=	32.768 Bit
path memory (trellis):	$(1632 \cdot 64) \cdot 16$	=	1.671.168 Bit
	$\sum$	=	1,9 Mbit

Table 2. Memory Requirements of the Viterbi Decoder

sented method can also be applied to any other Viterbi decoder where a termination is present in the encoding.

Furthermore, special features for the data path of our DSP were derived. Again, these features do not only favor Viterbi decoders but can also be used for faster computation of an FFT.

The described parallization method and special data path design allow us to perform the computation of a large Viterbi decoder on an application-specific DSP derived from a platform concept. The DSP will be able to perform signal processing for a complete DVB-T receiver which previously was only feasible for ASICs or large multi-processor systems.

7. REFERENCES

[1] Martin Bossert, *Kanalcodierung*, Teubner Verlag, Stuttgart, 1998.

[2] Gerhard P. Fettweis, *Parallelisierung des Viterbi-Decoders: Algorithmus und VLSI-Architektur*, Ph.D. thesis, RWTH Aachen, 1990.

[3] ETSI, *Digital Video Broadcasting (DVB); Framing Structure, channel coding and modulation for digital terrestrial television*, 2001, European Standard (Telecommunications Series).

[4] LSI Logic Inc., *L64782 Single Chip OFDM Receiver*, http://www.lsilogic.com/products/consumer/stb/index.html.

[5] Digilab 2000 Srl., "Professional DVB-T Receiver," Company Website.

[6] Matthias Weiss, Frank Engel, and Gerhard P. Fettweis, "A New Scalable DSP Architecture for System on Chip (SOC) Domain," in *Proceedings of ICASSP'99*, Phoenix, AZ, April 1999, vol. 4, pp. 1945–1948.