Scalable Reconfigurable Channel Decoder Architecture for Future Wireless Handsets

Gummidipudi Krishnaiah
Computer Science Department
Indian Institute of Technology
New Delhi, India
Krishna@cse.iitd.ac.in

Nur Engin, Sergei Sawitzki
NXP Semiconductors
High Tech Campus 31
5656 AE Eindhoven, The Netherlands
Nur.Engin@nxp.com, Sergei.Sawitzki@nxp.com

Abstract

The current trend in the consumer devices and communication service provider market is the integration of different communication standards within a single device (e.g. GSM phone with Bluetooth, WLAN and infrared interface) requiring tight integration of mobile broadcast, networking and cellular technologies within one product. Channel decoder is traditionally one of the most computationally intensive building block within digital receivers. The aim of this paper is to investigate the feasibility of a programmable channel decoder that can be dynamically reconfigured for decoding turbo and convolutionally encoded streams from various wireless standards. The architecture options are presented and the area costs and flexibility compared between the options. The resulting decoder architecture supports hardware resource sharing and reconfiguration between different standards and decoders and is more efficient in terms of silicon area than independent implementation of every decoder on the same IC.

1. Introduction

Many of the modern electronic consumer devices like PDAs, mobile and smart phones, set-top boxes etc. incorporate interfaces for different communications standards. A hand-held device connected to WLAN and GSM network and at the same using the Bluetooth link to headset for voice transfer is already a commodity. There are also products on the market or under development, which are capable of receiving and processing several data streams of the same standard (e.g. a set-top box with receiver, recorder and picture-in-picture feature). The trend towards multi-standard and multi-stream support will evolve further, requiring tight integration of mobile broadcast, networking and cellular technologies within one device. Channel decoding is one of the most computation intensive building blocks of a digital transceiver and many standards are using different codes or variations within the same code class. Even within one standard there are usually several coding schemes for different transmission and noise conditions. This diversity is the major problem in the integration of different channel decoders within one chipset. Taking into account the high costs of redesign steadily growing with silicon technology downscaling, there is a need for reconfigurable future-proof solution which can adapt to different standards depending on the use case as well as allowing a reconfiguration toward possible future standards (at least the ones using similar coding schemes). A careful look at the channel codes incorporated in many communication standards reveals that most of them are using convolutional and turbo codes. Although block sizes, polynomials and coding rates differ both within one coding scheme as well as between the coding schemes, it is possible to find common scenarios in both data path and memory usage to share them among different Viterbi and turbo decoders. At the same time such a multi-standard multi-steam decoder can be made scalable which is a very relevant issue for future-proofness, taking into account the continuously growing data rates.

2. System Requirements and Challenges

Table 1 summarizes the current and future standards where a convolutional or turbo code is (being) specified. Each standard has its own parameters for the channel encoder, such as different constraint lengths, code rates and generator polynomials and (in the case of turbo coding) different interleaving patterns. Thus, the channel decoder should support a wide range of coding parameters.

Viterbi decoders used for decoding convolutional codes require from less than 100 kbps up to 500 Mbps thorughput. Similarly, for turbo coded streams, the throughput ranges

978-3-9810801-2-4/DATE07 © 2007 EDAA

Table 1. Overview of the convolutional and turbo codes specified in wireless standards

Standard	Decoder	Maximum block size [bits]	Number of states	Mother code rate	Maximum user bit rate [Mbps]	Maximum workload [GOPS]
UMTS (R'99)	Viterbi	504	256	$\frac{1}{2};\frac{1}{3}$	0.064	0.07
CDMA2000	Viterbi	744	256	$\frac{1}{2}\ldots\frac{1}{6}$	0.038	0.039
DAB	Viterbi	none	64	$\frac{1}{4}$	1.109	0.28
DVB	Viterbi	1624	64	$\frac{1}{2}$	32	8
802.11a/g	Viterbi	32000	64	$\frac{1}{2}$	54	14
UWB	Viterbi	32768	64	$\frac{1}{3}$	480	123
GSM	Viterbi	876	16	$\frac{1}{2}$	0.0096	0.0006
GSM-EDGE	Viterbi	870	64	$\frac{1}{2}$	0.384	0.098
UMTS (R'99)	Turbo	5114	8	$\frac{1}{3}$	2.3	2.4
UMTS (HSDPA)	Turbo	5114	8	$\frac{1}{3}$	14	14
UMTS (LTE)	Turbo	5114	8	$\frac{1}{3}$	100	102
CDMA2000	Turbo	20730	8	$\frac{1}{5}$	2.4	2.5
802.16e	Turbo/8 iter	4800	8	$\frac{1}{3}$	20	21

from 64 kbps to 100 Mbps. Because of this large spread in the required throughput, a scalable channel decoder architecture is needed, making a redesign towards a lower or higher throughput simple.

3. State of the Art

Up to now, few approaches combining the turbo and Viterbi decoding have been reported. Bickerstaff et al have proposed a unified architecture designed for UMTS base stations [1, 2]. The main emphasis is on the multi-channel aspect, and the flexibility in coding schemes has not been handled in this work. Mainly the memories are shared between the turbo and Viterbi modes. In [3], another combined architecture is suggested for wireless terminals. In this architecture the datapath is shared as well as the memories. A MAP algorithm is used for decoding both convolutional and turbo codes. This is, however, only possible when the throughput requirement for Viterbi decoding is much lower than that of turbo decoding (e.g. 12.2 kbps for Viterbi and 384 kbps for turbo). In another effort to combine the two types of decoders soft Viterbi decoding is used for the turbo iterations and hard output Viterbi decoding is used for convolutionally encoded signals [4].

4. Architecture Considerations

The architectures for turbo and Viterbi decoding consist of two main elements: computation and storage. In this section, we investigate these parts separately.

4.1 Datapath Considerations

For both turbo and Viterbi decoding the main building block of the datapath is the so-called add-compare-select (ACS) operation. However, there are significant differences in the interconnect and dependencies between the ACS operations. For instance 802.11 a/g and UMTS HSDPA standards have different throughput but similar GOPS requirements (Table 1). The UMTS HSDPA turbo decoding is iterative and has only 8 states whereas the Viterbi decoding has 64 states each of which have to be processed only once. The datapath of turbo is less wide than that of Viterbi (8 vs. 64 or more states), but it has to run much faster because of the iterations (8 iterations assumed). In order to enable the turbo datapath to run at feasible clock speeds, multiple datapaths can be introduced, running multiple windows in parallel [7, 8]. For such parallel window architectures, both the intermediate data memory and interleaver address LUT need to have sufficient number of banks. Figures 1 and 2 show the area and power estimations based on [12] for parallel window versus a default multiple decoder architecture. A 200 Mbps decoder requires approximately 2.4 mm^2 for the multi-window case, whereas the same throughput using multiple decoders would require more than 7 mm^2 (90 nm CMOS technology).

4.2 Memory considerations

The memory requirement and data-organization are analyzed for both of the decoding schemes. In case of Viterbi

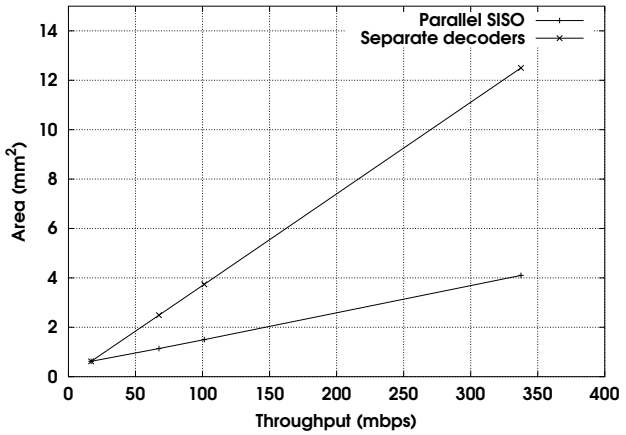

Figure 1. Area comparison of parallel window and multiple decoder turbo architectures.

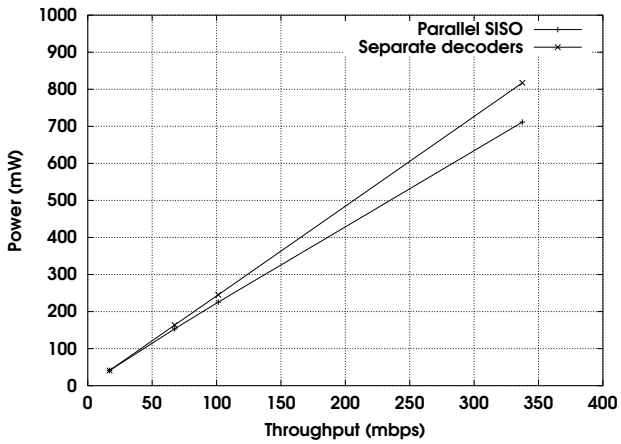

Figure 2. Power comparison of parallel window and multiple decoder turbo architectures.

decoder, the largest memory is the survivor memory. For a 256-state Viterbi in realistic channels this memory would require as much as 64 Kbits, with a bandwidth of 256 bits each cycle. The input and output buffer sizes required for the Viterbi decoder are not large, a 2–3 times the trace back depth is sufficient (at most 3 Kbits). Path metric memory is relatively small, at most 256 words of 8 bits are required.

For a turbo decoder, memory occupies more than 75% of the area. Turbo input buffer is up to 61 Kbits large for UMTS code blocks and the extrinsic memories are 70 Kbits. Furthermore, each turbo datapath includes some memories for intermediate data such as alpha metrics.

Memory sharing between the extrinsic turbo memory and the Viterbi survivor memory seems most favorable, as both the size and the bandwidth required for these memories are similar when multi window turbo datapath is used. Furthermore, the input buffers can be shared although this will not save much area as the Viterbi buffers are relatively small.

5. Reconfigurable Channel Decoding Architecture

The considerations for the datapath have been explained in the previous section. For sharing the same datapath for parallel-window turbo decoding and Viterbi decoding, we introduce a combined datapath with reconfigurable interconnect. This enables using the same datapath for 8 times 8-state turbo or once 64-state Viterbi decoding. Furthermore, we introduce the concept of a sliced architecture where each slice is a channel decoder sub-block containing 8-state trellis decoding hardware. The resulting architecture concept is shown in Figure 3. The reconfigurable interconnect also supports time-sharing same trellis decoding hardware, enabling Viterbi decoding for larger constraint sizes (i.e. more than 64 states). For example, 256-state Viterbi decoding at 1/4 of the throughput for 64-state decoding can be programmed using time-shared configurations.

Sharing of memories require extra configurable interconnects and control logic. Furthermore, a controller is included, which is basically a look-up-table where configuration information is stored for all supported standards. Once the decoding standard is known, this unit will generate control signals for all logic and interconnect at appropriate time intervals.

The functionality of the architecture is explained by considering two extreme cases of Viterbi and Turbo decoding. For Viterbi, 64-state WLAN requires high throughputs and 256-state UMTS needs huge data path for decoding. By using eight slices we realize a 64-state data-path, which can meet the high throughput requirements of WLAN. Since UMTS does not have high throughput requirements in case of convolution codes, we time-share the same 64-state datapath to realize a 256-state data-path. Thus it reduces the logic requirement by a factor of 4 with an overhead of increasing the interconnect flexibility. If higher throughput is required then this architecture can be easily scaled to accommodate more slices.

Note that at this abstraction level, it is still not decided which decoding algorithms should be supported by the datapath slices. Specifically, it is possible to implement turbo decoding by means of both Soft-Output Viterbi Algorithm (SOVA) and Maximum Aposteriori Probability (MAP) algorithms. Both of these algorithms consume soft (intrinsic) information input and produce soft (extrinsic) output. For convolutionally encoded streams, the standard decoding is

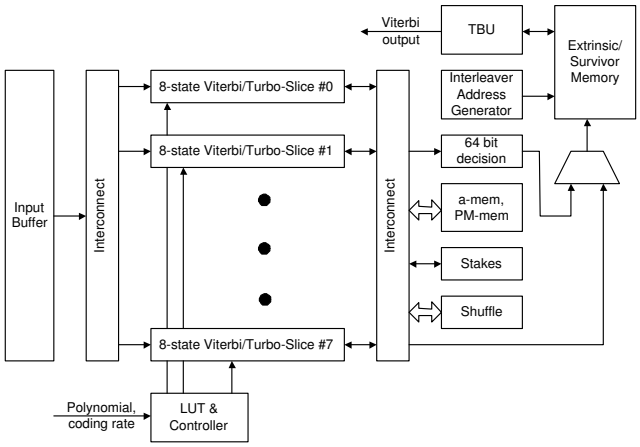

Figure 3. Reconfigurable Sliced Architecture for Multi-Standard channel decoding.

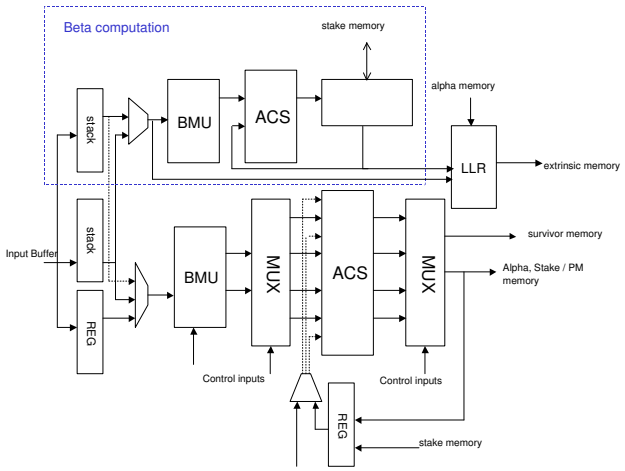

Figure 4. Internal Architecture of a Slice.

based on the Viterbi algorithm. However it is also possible to choose different decoding algorithm. Using the same algorithm for both turbo and convolutionally encoded streams has the advantage of simplicity, but also the combined data-path must satisfy the throughput and BER requirements for both cases.

5.1 Slice Architecture Exploration

We discuss variations of algorithm combinations that can be used to decode turbo and convolutionally encoded streams in this section. The aim is to choose a favorable tradeoff between error correction performance and area.

In the first architecture, we use Soft-Output-Viterbi-Algorithm (SOVA) for Turbo decoding and Hard Decision (HD) Viterbi for decoding convolution codes. In this architecture there are eight HD Viterbi Slices. One of the slices is extended to support 8-state SOVA for turbo decoding.

The second architecture uses suboptimal (Max Log or Max* Log) MAP decoding algorithm for both convolution and turbo decoding [9]. Each slice here can do 8-state MAP decoding. Thus for turbo, eight windows can be processed in parallel achieving 8 times higher throughput, as explained in section 4.1 and shown in [10].

In the third architecture, we make separate data-paths for 64-state HD Viterbi and 8-state suboptimal MAP but share the extrinsic and survivor memories. This architecture has drawbacks in terms of scalability and flexibility of the datapath: it is not possible to increase the turbo decoding throughput by employing multiple parallel windows, as the turbo datapath is only 8 states wide.

The fourth and fifth architectures employ sharing of the datapath at much finer grain. For this, the slice architecture shown in Figure 4 is introduced. As both MAP and Viterbi algorithms work on trellis modulation scheme, the decoding data-path logic (like BMU and ACS) of each slice is shared by making them more generalized to decode both convolution and turbo streams. The BMU generates all possible branch metrics, which are routed to ACS unit through a flexible interconnect. The ACS block can be configured to do add-compare-select or max operation depending on Viterbi or Turbo stream. To support various decoding requirements like generator polynomials, code rates, and constraint lengths in various standards, we build a flexible interconnect (cross bar) around the BMU and ACS blocks. The interconnect network is controlled by signals which are external to the slice.

We show two kinds of slices that can be chosen based on the throughput requirements. In the fourth architecture, the same data-path logic (BMU, ACS) is time-shared for both alpha and beta computation, thus requiring two cycles to produce one extrinsic value. Though this architecture works on 8 parallel windows, it effectively decodes 4 windows in parallel. It is the architecture with maximal logic sharing and hence minimal area.

In the fifth architecture alpha and beta computation is done in parallel, generating extrinsic value every clock cycle. The alpha computation data-path is the one that is reconfigurable and shared with convolution decoding. For beta computation extra logic is required per slice, but it delivers twice the throughput compared to the first case with smaller area overhead. This architecture can effectively decode 8 windows in parallel and is thus better suited for very high throughput needs.

The comparison between the five architecture options discussed, is shown in the Table 2. For reference, the last column includes seperate turbo and Viterbi decoders. The area estimations are based on earlier decoder implementa-

Table 2. Area comparison between reconfigurable sliced architectures

	Arch1 ML MAP + SOVA	Arch2 parallel ML MAP	Arch3 Viterbi + ML MAP	Arch4 time shared α and β	Arch5 parallel α & β	Seperate turbo and Viterbi decoders
HD Viterbi Slice	0.16	-	0.18	-	-	0.18
SOVA Slice	0.06	-	-	-	-	-
ML MAP Slice	-	0.84	0.10	-	-	0.84
Reconfigurable Slice 1	-	-	-	0.48	-	-
Reconfigurable Slice 2	-	-	-	-	0.80	-
Interleaver address LUT	0.14	0.23	0.14	0.23	0.23	0.23
Extrinsic/Survivor memory	0.18	0.30	0.18	0.30	0.30	0.42
Input buffer turbo	0.19	0.19	0.19	0.19	0.19	0.19
Input buffer Viterbi	0.03	0.03	0.03	0.03	0.03	0.03
Stake memory	0.04	0.52	0.04	0.13	0.13	0.13
Alpha memory	0.02	0.60	0.02	0.15	0.15	0.15
Viterbi path metric registers	0.03	-	0.03	0.03	0.03	0.03
Trace back unit	0.13	-	0.13	0.13	0.13	0.13
Interconnect / Control logic	0.02	0.02	0.02	0.02	0.02	0.02
Total area (mm^2)	1.00	2.73	1.07	1.69	2.01	2.35
Throughput $K = 7$ Viterbi (Mbps)	300	150	300	300	300	300
Throughput $K = 4$ turbo (Mbps)	15	120	15	60	120	120
Turbo performance compared to LogMAP (dB)	0.8	0.3	0.3	0.3	0.3	0.3

tions [7, 12, 13], assuming 90 nm CMOS technology. For throughput calculations, a clock speed of 300 MHz is assumed. The performance of SOVA and suboptimal MAP algorithm options have been verified by means of algorithm simulations. The first three architectures use (combinations of) SOVA or suboptimal MAP algorithms. In the first architecture, by using SOVA for turbo, more logic can be shared with HD Viterbi, thus saving in area [7, 11]. However, SOVA is not a better choice if the BER requirement is of order 10^{-6} or less, as it hits the noise floor earlier than maximum-a-posteriori (MAP) based algorithms. Alternatively, in Architecture 2 MAP algortihm is used for both code types. By using MAP instead of Viterbi, the complexity of convolution decoding increases by more than a factor of 4 [5, 6]. This makes the architecture an inefficient one, if high throughput convolution decoding is considered.

In Architectures 1 and 3 no parallel turbo windows are used, leading to lower throughput but also lower memory area. The other architectures have the same memory capacity but higher memory area because of multiple-bank memory architecture leading to memories less efficient in area. When throughput requirements for turbo decoding is not very high, a 64-state Viterbi datapath and an 8-state turbo datapath sharing same extrinsic/survivor memory (Architecture 3) is a good choice.

Architectures 4 and 5, where the slice architecture given in Figure 4 is used, are good choices when high flexibility and high turbo throughput is required. The throughputs for seperate decoders and Architecture 5 are the same. However, the seperate decoders have higher throughput for the multi-channel case, since both turbo and Viterbi streams can be decoded simultaneously. These two architectures are also best in terms of scalability, since all slices have the same architecture.

The results show that there is not much area advantage gained due to datapath sharing. In Architectures 4 and 5 the area of the datapaths are very close to the case of other architectures when normalized for the same throughput. The most significant gain in area compared to the seperate decoder case is achieved by sharing the survivor/extrinsic memories. Mainly due to the sharing of memories, the Architectures 4 and 5 have an area advantage of 27% and 17% respectively, compared to using seperate (flexible) turbo and Viterbi decoders.

Comparison of area and throughput estimates with other combined Viterbi-turbo architectures yields favorable results. One example is the unified decoder described in [1]. Designed for 18 nm CMOS technology, this decoder has an area of 9 mm^2 and can decode 2 Mbps turbo plus 128 kbps of 256-state Viterbi for UMTS voice channels, operating at clock frequency of 100 MHz. When scaled to 90 nm CMOS technology, the area figure for this decoder becomes 2.25 mm^2. Our decoder architecture is estimated to have an area under 2.35 mm^2, and, if clocked at 100 MHz, can decode k=4 Turbo at 20 Mbps plus 256-state Viterbi at 12 Mbps. Furthermore, we are aiming at a multi-standard architecture (which is taken into account in area estimations) while the decoder mentioned in [1] is for UMTS turbo and Viterbi decoding only.

6. Conclusions and Future Work

An architecture concept for a reconfigurable channel decoder has been introduced. For wireless handsets, the application space is very large and aiming to support all applications with one decoder will lead to over-dimensioning. Instead, a scalable approach has been adopted in this work, which has led to a decoder architecture that can support different application scenarios and also enables fast scaling/redesign.

Architecture exploration has been conducted to verify the possible tradeoffs in throughput, area and BER. The lower BER performance of SOVA and high complexity of MAP make using same algorithm for both turbo and Viterbi an unfavorable option. The area results for architectures combining MAP with HD Viterbi show that sharing the datapaths does not deliver much area advantage. The best way to save area is sharing the survivor memory for Viterbi with the extrinsic data memory for turbo. This is especially true when using multi-window turbo, since the bandwidths required for both of these memories fit very well in this case.

An executable model of the described architecture has been implemented in System C. Future work will focus on the estimation of power consumption and evaluation of scenarios for high speed, low-power decoding.

References

[1] Bickerstaff, M. A., Garrett, D., Prokop, T., Thomas, C., Widdup, B., Zhou, G., Davis, L. M.: A Unified Turbo/Viterbi Channel Decoder for 3GPP Mobile Wireless in 0.18-mm CMOS, IEEE Journal of Solid-State Circuits, November 2002, pp. 1555–1564

[2] Thomas, C., Bickerstaff, M. A., Davis, L. M., Prokop, T., Widdup, B., Zhou, G., Garrett, D., Nichol, C.: Integrated Circuits for Channel Coding in 3G Cellular Mobile Wireless Systems, IEEE Communications Magazine, August 2003, pp. 150-159.

[3] Kreiselmaier G., Vogt T., Wehn N.: Combined Turbo and Convolutional Decoder Architecture for UMTS Wireless Applications, Proceedings of DATE, February 2004, pp 192-197.

[4] Cavallaro, J. R., Vaya, M.: VITURBO: A Reconfigurable Architecture for Viterbi and Turbo Decoding, Proceedings of ICASSP '03, April 2003, pp. 497–500

[5] Robertson P., Villebrun, E. and Hoeher, P.: A Comparison of Optimal and Sub-optimal MAP decoding algorithms operating in Log domain, Proc. IEEE ICC, Seattle, WA, USA, Jun 18-22, 1995, pp. 1009-13.

[6] Chatzigeorgiou, I. A., Rodrigues, M. R. D., Wassell, I. J., and Carrasco, R.: A Comparison of Convolutional and Turbo Coding Schemes for Broadband FWA Systems, 12th International Conference on Telecommunications, Cape Town, South Africa, May 2005.

[7] Joeressen O. J., and Meyr, H.: 40 Mb/sSoft-Output Viterbi Decoder, IEEE Journal of Solid State Circuits, vol.30 July 1995, pp. 812-818.

[8] Engin, N.: Turbo decoder architecture with scalable parallelism, in Proceedings of IEEE Workshop on Signal Processing Systems, 2004, pp. 298–303

[9] Berns, F., Kreiselmaier, G., Wehn, N.: Channel Decoder Architecture for 3G Mobile Wireless Terminals, Proceedings of DATE Conference, February 2004, pp. 192–197

[10] Yoon S., Bar-Ness Y.: A parallel MAP algorithm for low latency turbo decoding, Communication Letters IEEE, vol.6 Jul 2002, pp.288-290.

[11] Bekooij, M., Dielissen, J., Harmsze, F., Sawitzki, S., van der Werf, A., van Meerbergen ,J.: Power-Efficient Application-Specific VLIW Processor for Turbo Decoding, in Proceedings of the IEEE International Solid-State Circuits Conference (ISSCC'2001), February 2001

[12] Harmsze, F., Dielissen, J.: Implementation of a Turbo Decoding for 3G — a turbo decoding processor, Nat.Lab. Technical Report 7178, Philips Research, 2001

[13] Bi, L., Pu, T., Sawitzki, S.: Architecture Study of 480 Mbps Viterbi Decoder for OFDM-UWB: Towards implementation in silicon, Nat.Lab. Technical Note PR-TN 2004/00779, Philips Research, 2004