14th European Signal Processing Conference (EUSIPCO 2006), Florence, Italy, September 4-8, 2006, copyright by EURASIP

HIGH-PRECISION LDPC CODES DECODING AT THE LOWEST COMPLEXITY

Massimo Rovini, Francesco Rossi, Nicola E. L'Insalata and Luca Fanucci

Dept. of Information Engineering, University of Pisa
via G. Caruso, I-56122, Pisa, Italy
email: {*massimo.rovini, francesco.rossi, nicola.linsalata, luca.fanucci*}@iet.unipi.it

ABSTRACT

This paper presents a simplified, low-complexity check node processor for a decoder of LDPC codes. This is conceived as the combination of the modified Min-Sum decoding with the reduction of the number of computed messages to only $P+1$ different values. The simulations with a random code used as a case study show that this technique performs excellently even when only two different values are propagated ($P=1$). This result is assumed as the basement to the design of an optimised serial architecture. The logic synthesis on $0.18\,\mu$m CMOS technology shows that our design outperforms in complexity similar state–of–the–art solutions and makes the check node operations no longer critical to the complexity of the whole decoder.

1. INTRODUCTION

In the 'post-turbo-codes' era of forward error correction history, low-density parity-check (LDPC) codes, first introduced by Gallager in 1963 [1], have gained a big momentum from academia and industry. Thus, LDPC codes have started to play a crucial role in modern communication systems, which are highly demanding for both performance close to the Shannon limit and very-high data rate services.

After their rediscovery in 1995 [2], several research activities have aimed at implementing low-complexity and high-throughput decoders [3]. Indeed, despite the recent advances in microelectronics technology, the main issue in the decoder design is still to keep the circuit complexity under control; this trouble especially arises in the design of decoders for long block codes, as those of DVB-S2 [4], or for high data rate applications, as WLAN (802.11n) and WMAN (802.16e). High-speed decoding is usually achieved by hardware replication, paid at the cost of an increased complexity and chip routing congestion.

One route followed to solve the problem has been the "decoder–first" [5] or "architecture–aware" design [6], [7], in which the decoder architecture is first conceived in such a way to simplify the parallel processing at high data rate. Then, the error correction performance of the so-defined code is somehow assessed a posteriori. However, the need of very low complexity elaborations is still an appealing issue. To this extent, several approximations of the original Belief Propagation (BP) algorithm have been proposed in the past [8], [9], which are very helpful to define optimised VLSI architectures [10].

Neglecting the contribution of the communications between check and variable node processors (CNP, VNP), which basically consists in routing congestion and influences the data management in the *extrinsic* memory, the decoder complexity has always been concentrated on CNPs rather than VNPs, either in serial [10], [11] or in parallel implementations [12]. In [13] a simplified CN update is described, in which only two magnitudes are output by the processor, independently of the node degree. As a result, noticeable savings in memory (less locations need to be stored) and chip area (simpler operations are performed) are achieved.

Starting from this observation, this paper combines the 2-output algorithm [13] with the modified Min-Sum decoding (mod-MS) described in [9] to push the CNP complexity at its minimum. To this aim, the 2-output rule is first generalised to $P+1$ outputs. A similar strategy is adopted by [14], where only the $P=\lambda$ less reliable input magnitudes are used to simplify the operations on check nodes, the remaining ones being ignored. However, it is shown in [14] that $\lambda=3$ must be used at least for optimal performances.

After the analysis of the error correction performance, a very low complexity implementation of the CNP is detailed. The proposed architecture allows savings in area so high to make the VNP, and no longer the CNP, dominate the overall decoder complexity.

2. LOW–COMPLEXITY LDPC DECODING

Decoding of LDPC codes is carried on as an iterative exchange of messages along the Tanner graph underlying the LDPC code, aiming at refining the MAP estimation of the transmitted bits.

To favour an efficient implementation, signs and magnitudes are separately updated on check nodes (CN). If ε_{ij} denotes the message from CN i to VN j, and μ_{ij} the message in the opposite direction, the CN update on signs is performed according to the exact rule:

$$-\operatorname{sign}(\varepsilon_{ij}) = \prod_{k\in\mathcal{N}(i)\setminus j} -\operatorname{sign}(\mu_{ik}) \tag{1}$$

with $\mathcal{N}(i)$ the set of VNs connected to CN i.

On the contrary, for the lowest implementation complexity, magnitudes update may resort to the mod-MS [9] algorithm, based on the binary operator M-min*. This is defined as M-min$^*(|a|,|b|) \doteq \min(|a|,|b|) - \log\left(e^d/1+e^d\right)$, with $d=||a|-|b||$. This operator was already proven to perform excellently from the implementation loss point of view [15]. As a further simplification, messages are updated according to a genaralisation of the strategy described in [13]. The resulting update rule of magnitudes is:

$$|\varepsilon_{ij}| = \begin{cases} \text{M-min}^*{}_{k\in\{\mathcal{N}(i)\setminus j\}}\left(|\mu_{ik}|\right), & j\in\mathcal{N}_P(i) \\ \text{M-min}^*{}_{k\in\mathcal{N}(i)}\left(|\mu_{ik}|\right), & j\notin\mathcal{N}_P(i) \end{cases} \tag{2}$$

where $\mathcal{N}_P(i)$ is the subset with cardinality $P=|\mathcal{N}_P(i)|$ of the input messages in $\mathcal{N}(i)$ with smallest reliability.

Equation (2) means that only $P+1$ different magnitudes are output by the processor: a dedicated, exact value in response to any message in $\mathcal{N}_P(i)$, plus a generic value common to all input messages not in $\mathcal{N}_P(i)$ and computed on the whole set of inputs. When $P = |\mathcal{N}(i)|$, exact mod-MS takes place, while for $P = 1$ we get the M-min* version of the algorithm described in [13].

A similar approach is used in [14], which restricts to the messages in $\{\mathcal{N}_P(i) \backslash j\}$ for the computation of the $\lambda = P$ dedicated magnitudes, $j \in \mathcal{N}_P(i)$. Accordingly, the remaining messages not in $\mathcal{N}_P(i)$ are updated with a common magnitude computed on the same $\mathcal{N}_P(i)$ instead of $\mathcal{N}(i)$ as in (2). Moreover, no approximations of the binary operator, but exact log-BP is considered in [14].

On VNs, check–to–variable messages are combined together and summed to the so-called channel *intrinsic* information, as usual. The latter is also referred to as *a priori* information, which stems from the channel demodulator in the form of a *Log-Likelihood Ratios* (LLR), denoted as λ.

$$\mu_{ij} = \lambda_j + \sum_{k \in \mathcal{M}(j) \backslash i} \varepsilon_{kj} \tag{3}$$

In (3), $\mathcal{M}(j)$ is the set of CNs connected to VN j. The algorithm is initialised with $\mu_{ij} = \lambda_j$ for any $j = 0, 1, ..., N-1$, and proceeds iteratively until all parity checks are verified or a maximum number of iterations is reached.

3. CHECK NODE PROCESSOR ARCHITECTURE

The CNP has been designed for a serial I/O (input/output) data stream. Compared with a parallel architecture, the serial solution offers an inherent lower complexity, as only a single M-min* operator is enough, in principle[1]; easier reconfigurability is achieved by dimensioning the processor on the maximum degree (worst case) and properly driving nodes with smaller degree; hardware resources are not wasted with irregular codes; last but not least, it is a compulsory choice for the lowest-complexity implementations of (1) and (2). Indeed, on the one hand a parallel CN processor would need a tree of binary operators M-min*, on the other hand it would make the re-ordering of the input messages impractical.

The principle architecture of a CNP computing $P+1$ different magnitudes is sketched in Fig. 1. Here, signs are updated in the CN *Sign Processor* while magnitudes are updated in a dedicated section. The former, implementing (1), is detailed in Fig. 2. The latter basically composes of an *Input Forming* stage, which reorders the incoming stream of magnitudes in such a way to put on the last positions those P values with the lowest reliability; then, a M-min* accumulator designed as detailed in [10] is used to compute the outputs of the processor; eventually, a multiplexer assigns one of the $P+1$ magnitudes to each edge according to (2). Here, output magnitudes are labelled with ϑ_i, $i = 0, 1, ..., P$, index P denoting the un-marginalised value, i.e., $\vartheta_P = \text{M-min}^*_{k \in \mathcal{N}(i)} \{|\mu_{jk}|\}$.

The *Control Unit* (CU) of Fig. 1 is in charge of supervising the whole data-flow of operations. Indeed, the CU itself is controlled by a 1-bit synchronisation strobe, which signals the first message of a new elaboration. The use of a single-bit

[1]Despite this borderline case, serial implementations usually rely on more than one operator, which would be detrimental to the decoding throughput. In [16] three operators are proposed for better performance.

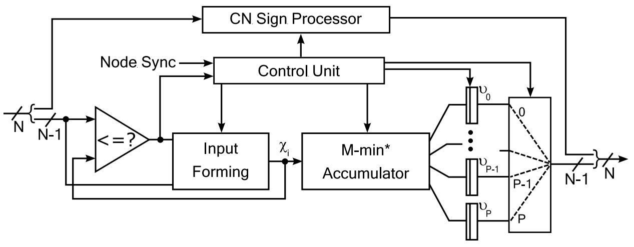

Figure 1: Check node processor: top level serial architecture.

control signal prevents risks of crowding in the chip routing, and does not add any significant hardware overhead, at the same time.

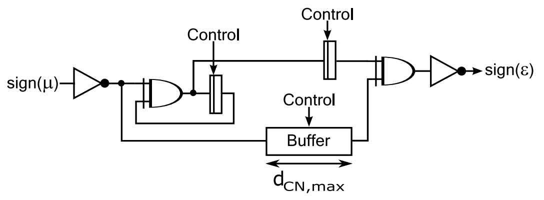

Figure 2: Check node: sign processor architecture.

The output multiplexer in Fig. 1 performs kind of decompression of the internal metrics onto the output *extrinsic* messages, by passing from $P+1$ to d_{CN} values. The elimination of such a component would be highly desirable, as messages could be stored in the compressed form and decompressed on–the–fly on use, thus saving memory locations. In practice, such a solution is only viable in layered decoders [17], or whenever the VN phase is skipped [18]. In this case, the only drawback is that routing is roughly P times more congested, especially in a replicated design, as processors must handle a compressed input that roughly span $P+1$ regular messages. On the contrary, the presence of the VN phase as in a flooding decoder, would change the compressed vector retrieved from memory into an uncompressed one, thus making the memory compression inefficient.

Note that for a given P, the 1-bit delay line used for signs in Fig. 2 is the only element growing linearly in size with the CN degree. All the other elements, if not constant, vary as the logarithm of the degree.

The proposed architecture becomes inefficient for $P > 3$ and, in this case, alternative architectures such as the forward–backward recursion are preferable [10].

3.1 The 2-magnitude Architecture

The general architecture of Fig. 1 can be optimised to achieve the highest efficiency in terms of both gate complexity and latency for $P = 1$. In this case, the two magnitudes can be computed with the architecture of Fig. 3.

Let d_{CN} be the CN degree and let χ_i, $i = 0, 1, ..., d_{CN} - 1$ be the input sequence after input forming, i.e. such that $\chi_{d_{CN}-1} \leq \chi_k$ $\forall k = 0, 1, ..., d_{CN} - 2$ is the less reliable magnitude. Then, the CNP would first calculate ϑ_0 through repeated applications of the M-min* operator on inputs $\chi_0, ..., \chi_{d_{CN}-2}$, in sequence; next, ϑ_1 is computed as $\vartheta_1 = \text{M-min}^* \left(\vartheta_0, \chi_{d_{CN}-1}\right)$. To this extent, the *Input Forming* stage is simply made up of a two-way multiplexer plus the MIN register which contains the less reliable input $\chi_{d_{CN}-1}$. The M-min* accumulator is composed of a single operator properly controlled by the CU. Also the output selector simplifies into a two-way multiplexer, whose inputs are ϑ_0 in correspondence of the less reliable input, or ϑ_1 otherwise.

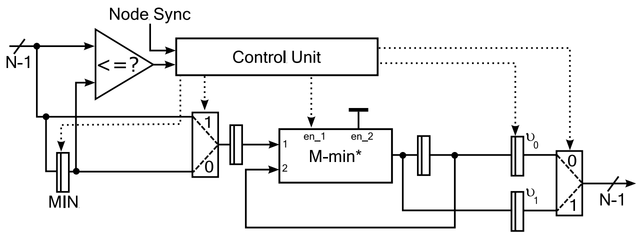

Figure 3: Check node: detailed architecture for P=1.

Once CNs have been ordered for increasing degrees to minimise the overall latency, the architecture of Fig. 3 and more generally of Fig. 1 allow the continuous, pipelined, elaboration of the input message stream.

4. RESULTS

4.1 Error Correction Performance

The performance of the technique presented in Section 2 has been assessed by simulation with a 4-cycle free random code, drawn from the D. MacKay's on-line database [19]. The code is labelled 4095.738.4.102, is quasi-regular with rate 0.82 and codeword size $N = 4095$.

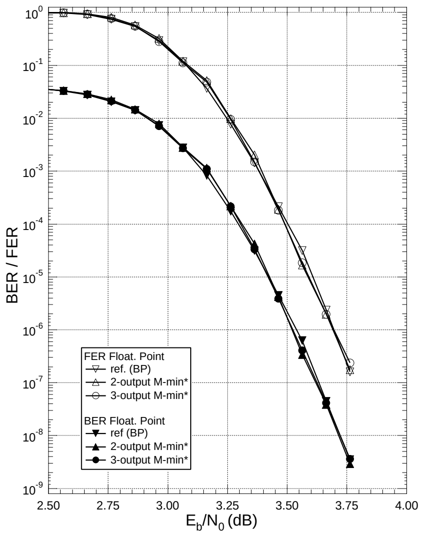

Figure 4: Floating point performance (4095.738.4.102, rate 0.82, BPSK, AWGN channel).

As discussed in Sections 2 and 3, higher values of P yield better performance, but also higher complexity. Taking the reduction of the decoder complexity as a priority, we tailored our simulations toward low values of P. To this extent, Fig. 4 reports the BER and FER performance over AWGN channel with BPSK modulation, of the exact message-passing algorithm (or belief-propagation, BP) and of the low-complexity approximation described in Section 2 for $P = 1$ (2-output M-min*) and $P = 2$ (3-output M-min*). Full precision, floating point, decoding is run in both cases. While the M-min* operator alone has already been proved to work excellently, no appreciable losses arise in performance down to FER $= 10^{-7}$ or BER $= 10^{-9}$ even with the lowest complexity approximation ($P = 1$).

The impact of finite-precision decoding has been investigated with the adoption of a quantisation scheme as detailed in [15]. Fig. 5 compares the BER curves of mod-MS decoding[2] (full markers) and of the 2-output M-min* approximation (empty marks) for two quantisation cases: a high-precision configuration with input LLR quantised on 6 bits and *extrinsic* messages on 7 bits, and a low-complexity solution with both signals on 5 bits. While no significant differences are observable with messages on 7 bits, surprisingly, the proposed 2-output M-min* approximation compares favourably when messages are on 5 bits. In this case, mod-MS decoding roughly pays 0.25 dB to the curve with infinite precision, but this gap shrinks to about 0.2 dB for our technique.

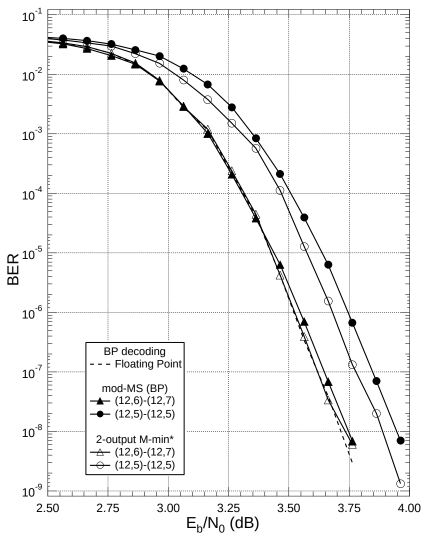

Figure 5: Bit-true performance (4095.738.4.102, rate 0.82, BPSK, AWGN channel).

4.2 VLSI Implementation

The architecture detailed in Section 3 has been synthesized for $P = 1$ with high–speed standard cells library on 0.18 μm CMOS technology and operating conditions 1.55 V and 85 C. The related complexity of the CNP, measured in equivalent gates[3], is shown in Fig. 6 as a function of the maximum node degree of the code, $d_{CN,\max}$ and with the *extrinsic* message width N_m as a parameter.

As far as complexity is concerned, hardware resources are employed very efficiently compared with the forward–backward architecture of [16] (labelled α-β in Fig. 6); for a given N_m, the latter is about 2 to 6.5 times more complex than the proposed solution. In addition, while complexity increases linearly with d_{CN} in [16], in the new technique it

[2]We use here mod-MS as a proxy of exact belief-propagation, as it is proven to perform excellently but with higher stability in fixed point domain.

[3]The reference unit for gate counting is the 2-input NAND gate with area 12.88 μm^2 on 0.18 μm CMOS technology.

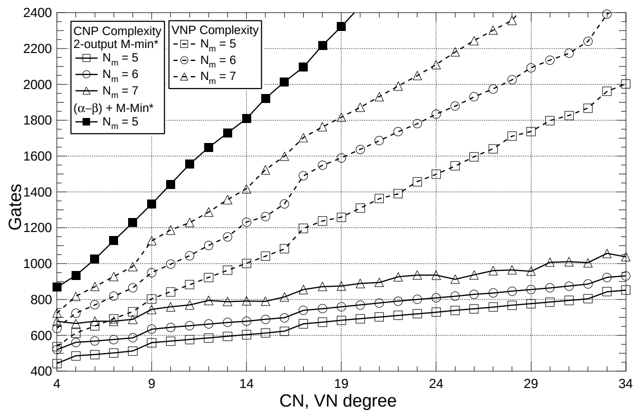

Figure 6: Complexity in equivalent gates of the proposed CNP architecture compared with [16] and the serial VNP with the message width as a parameter.

grows with the logarithm of d_{CN}: an increment of three times of $d_{CN,\max}$ (from 7 to 21), only increases the gates count of about 40%.

Concerning the timing, the latency of the proposed architecture is $d_{CN}+2$ clock cycles, while the synthesis reveals a maximum clock frequency always greater than 200 MHz, even in the worst case of $d_{CN}=34$ and $N_m=7$.

Along with the synthesis results of the CNP, Fig. 6 also reports the complexity of a serial VNP for the same message widths. Surprisingly, depending on the code, the VNP can even dominate the whole complexity!

The design of a decoder for DVB-S2 is described in in [11]. Here, serial functional units encompassing VNP and CNP are used, which feature 10.8 mm^2 overall or 4957 gates each[4]. In the same conditions we registered 865 gates for the CNP ($d_{CN,\max}=30$) plus 1150 gates for the VNP ($d_{VN,\max}=13$) with messages on 6 bits. This leads to considerable savings in complexity, about 59% of the area declared in [11].

5. CONCLUSION

This paper has proposed a low-complexity approximation of the elaborations performed on the check node of a decoder for LDPC codes. The offered solution combines the modified Min-Sum algorithm with the reduction of the messages computed by the CN processor to only $P+1$ different values. The two approximations have been demonstrated to coexist excellently, with no appreciable loss in performance compared to the exact message passing algorithm, even when $P=1$ and only two different magnitudes are delivered by the processor. Surprisingly, the quantisation of messages on few bits (5) even improves the error correction performance of our solution.

This is in line with the results of [13] where 2-output, not-approximated decoding is considered and outperforms [14], where the restriction of the messages used for update to only a subset of the whole, results into the need of $\lambda=P=3$ different values, at least, to get negligible losses. However, the enlargement to the whole set of input messages required by our technique does not add any complexity overhead, neither increase the computation latency of [14].

Along with the formal description of the elaborations, a generic architecture of the CN processor for use with small values of P has been discussed and optimised for the lowest complexity solution with $P=1$. The logic synthesis on 0.18 μm CMOS technology showed that the proposed architecture compares favourably with other state–of–the–art implementations. Remarkably, the design is so excellent that, depending on the code in use, the VNP can even dominate the complexity of the whole decoder. For instance, this is the case of a decoder for DVB-S2, where the adoption of the approximation described here would save about 59% of the complexity of the node processors (VN and CN, together) declared by similar works.

[4]In [11], 360 functional units are synthesised on 0.13 μm CMOS technology. For gate counting, we used 6.052 μm^2 for the area of the 2-input NAND.

Acknowledgment

This work was partially supported by the Italian Department of Education, University and Research (MIUR) under the Project FIRB PRIMO and by the European Commission under the Network of Excellence in Wireless Communications NEWCOM.

REFERENCES

[1] R. Gallager, “Low-Density Parity-Check Codes,” Ph.D. dissertation, Massachusetts Institutes of Technology, 1960.

[2] D. MacKay and R. Neal, “Good codes based on very sparse matrices,” in *5th IMA Conference on Cryptography and Coding*, Springer-Verlag, Ed., 1995, pp. 100–111.

[3] A. Blanksby and C. Howland, “A 690-mW 1-Gb/s 1024-b, rate-1/2 low-density parity-check code decoder,” *IEEE J. Solid-State Circuits*, vol. 37, no. 3, pp. 404–412, Mar 2002.

[4] “Digital video broadcasting (DVB); second generation framing structure, channel coding and modulation systems for broadcasting, interactive services, news gathering and other broadband satellite applications,” ETSI, June 2004.

[5] E. Boutillon, J. Castura, and F. Kschischang, “Decoder-First Code Design,” in *2nd International Symposium on Turbo Codes and Related Topics*, Sep 2000, pp. 459–462.

[6] M. Mansour and N. R. Shanbhag, “High-throughput LDPC decoders,” *IEEE Trans. VLSI Syst.*, vol. 11, no. 6, pp. 976–996, Dec 2003.

[7] H. Zhong and T. Zhang, “Design of VLSI implementation-oriented LDPC codes,” in *International Symposium on Low Power Electronics and Design*, vol. 1, Oct 2003, pp. 670–673.

[8] M. Fossorier, M. Mihaljevic, and H. Imai, “Reduced complexity iterative decoding of low-density parity check codes based on belief propagation,” *IEEE Trans. Commun.*, vol. 47, no. 5, pp. 673–680, May 1999.

[9] F. Zarkeshvari and A. Banihashemi, “On implementation of min-sum algorithm for decoding low-density parity-check (LDPC) codes,” in *Proc. IEEE GLOBECOM*, vol. 2, Nov 2002, pp. 1349–1353.

[10] M. Rovini, N. L'Insalata, F. Rossi, and L. Fanucci, "VLSI Design of a High-Throughput Multi-Rate Decoder for Structured LDPC Codes," in *8th Euromicro Conference on Digital System Design (DSD)*, Aug-Sept. 2005, pp. 202–209.

[11] F. Kienle, T. Brack, and N. Wehn, "A synthesizable IP core for DVB-S2 LDPC code decoding," in *Design, Automation and Test in Europe (DATE)*, 2005, pp. 100–105.

[12] G. Masera, F. Quaglio, and F. Vacca, "Finite precision implementation of ldpc decoders," in *IEE Proc. Commun.*, vol. 152, Dec 2005, pp. 1098–1102.

[13] C. Jones, E. Valles, M. Smith, and J. Villasenor, "Approximate-MIN Constraint Node Updating for LDPC Code Decoding," in *IEEE Military Communications Conference (MILCOM)*, vol. 1, Oct 2003, pp. 157–162.

[14] F. Guilloud, E. Boutillon, and J. Danger, "λ-Min Decoding Algorithm of Regular and Irregular LDPC Codes," in *Proc. 3rd International Symposium on Turbo Codes & Related Topics*, Sept 2003.

[15] M. Rovini, N. L'Insalata, F. Rossi, and L. Fanucci, "LDPC Decoding in Fixed-Point Precision: a Systematic Quantisation Study," in *International Conference on Software, Telecommunications and Computer Networks (SoftCOM)*, 2005.

[16] X.-Y. Hu, E. Eleftheriou, D.-M. Arnold, and A. Dholakia, "Efficient implementations of the sum-product algorithm for decoding LDPC codes," in *Proc. IEEE GLOBECOM*, vol. 2, Nov 2001, pp. 1036–1036E.

[17] D. Hocevar, "A Reduced Complexity Decoder Architecture via Layered Decoding of LDPC Codes," in *IEEE Workshop on Signal Processing Systems, SISP 2004*, 2004, pp. 107–112.

[18] M. Castano, M. Rovini, N. L'Insalata, F. Rossi, R. Merlino, C. Ciofi, and L. Fanucci, "Adaptive Single Phase Decoding of LDPC Codes," in *International Symposium on Turbo Codes and Related Topics*, Apr 2006, p. to appear.

[19] "Encyclopedia of Sparse Graph Codes," David MacKay's Research group, Cavendish Laboratory, *http://www.inference.phy.cam.ac.uk/mackay/*.