An Efficient Incremental Redundancy Implementation for 2.75G Evolved EDGE

Benjamin Weber, Harald Kröll, Christian Benkeser, Qiuting Huang

Integrated Systems Laboratory
ETH Zurich, 8092 Zurich, Switzerland
e-mail: {weberbe,kroell,benkeser,huang}@iis.ee.ethz.ch

***Abstract*—Incremental Redundancy (IR) in GSM was introduced with EDGE and later adopted in Evolved EDGE in order to increase the throughput. The distribution of IR controlling and processing over various parts of the PHY and RLC/MAC layer demands high loads to processing units and imposes difficulties with respect to system design. In particular, large amounts of data have to be moved between memories and processing units. In this work IR is implemented as part of baseband signal processing in order to disburden RLC/MAC layer processing units from IR. The open source baseband framework MatPHY is extended for packet switched operation to facilitate the development of efficient IR hardware architectures. With the design parameters obtained from performance evaluations, computed with MatPHY, we implemented an architecture in 130 nm CMOS technology to prove the suitability of the approach.**

I. Introduction

After its first release in 1990, GSM was fast becoming one of the most successful and widely used cellular standards worldwide. Although initially supporting circuit switched connections only, there was a need for packet switched operation. Correspondingly, GPRS and later EDGE (also referred to as EGPRS) services were introduced by the 3GPP standardization organization. Together with EDGE, additional modulation and coding schemes were introduced to maximize throughput. Irrespective of 3rd generation and LTE deployments, the GSM standard and its enhancements will be in use for many years to come. Bearing this in mind, the 3GPP introduced an enhancement to EDGE called Evolved EDGE (also referred to as EGPRS2) with up to 32QAM modulation and turbo codes. In order to keep the throughput at an acceptable level even under severe radio conditions, Incremental Redundancy (IR) operations were first introduced with EDGE and inherited by Evolved EDGE. IR is also referred to as Type II Hybrid ARQ. More detailed, if decoding of a data block proves unsuccessful, the encoded soft values (log-likelihood ratios) of the data block in question are kept at the receiver and a retransmission is requested. The retransmitted data block contains additional redundancy information. The receiver combines the retransmission with the previous transmission before a decoding attempt is started. Correspondingly, the coding rate decreases with each retransmission whereas the decoding success rate increases. Thus, IR aids to achieve high average throughput. Minimum average throughput requirements from the 3GPP specifications ask a Mobile Station (MS) to store up to 1024 Radio Link Control (RLC) blocks [1], [2]. Therefore, memory and processor challenges are two major concerns when implementing IR at MS side.

A hardware architecture of a MS consists of several processing components attached to a RF transceiver. The system processor for higher layer operations such as RLC/MAC is connected to memories and other peripherals. For digital baseband processing in the PHY a DSP with a number of accelerators is employed. Alternatively, there exist approaches (e.g. [3]) which make do without a DSP. Instead, the digital baseband processing is implemented in dedicated hardware only. According to literature (e.g. [4]), IR implementation is traditionally distributed over various processing components in a MS. Erroneous RLC blocks are stored in an external RAM. IR control processes run on the system processor. The decoding of the RLC blocks takes place in the digital baseband on a DSP's accelerator.

In such a conventional setup the IR mechanism claims high demands for the system processor, which is not desired. And, RLC/MAC software developers need to incorporate IR operations. Furthermore, the data bus between external memory, system processor and digital baseband can experience high loads due to IR processing. For emerging applications that require ultra low-cost and low-power devices, such as Internet of Things (IoT) or machine to machine (M2M) communication, which may use the GSM/EDGE standard due to its ubiquitous coverage, high computational loads are prohibitive. An IR processing block incorporated into the digital baseband comprising all IR related operations can remove IR complexity and load from the system processor and external components.

Contribution: In this paper IR implementation challenges are investigated and an efficient dedicated hardware architecture, which hides the IR complexity from the system processor and higher protocol layers, is proposed. In order to gain insight into the IR mechanisms, an Evolved EDGE capable version of the open source MatPHY GSM framework [5] was equipped with IR functionality. The minimum IR memory size in order to meet the 3GPP throughput requirements is determined based on the receiver performance of the extended framework

and a model RF front-end. The implemented IR hardware architecture using a dedicated on-chip memory for the storage of RLC blocks comprises depuncturing, memory management, and a soft information combiner unit. A corresponding VHDL model is implemented and synthesized.

Outline: The rest of this paper is organized as follows: Section II explicates IR operations and GSM/EDGE architectures in more detail. Subsequently, in Section III the MatPHY extensions are outlined followed by IR performance evaluations in Section IV. Afterwards, a dedicated hardware architecture is proposed followed by concluding remarks.

II. IR in GSM/EDGE MS Architectures

Packet switched connections use so called radio blocks consisting of a RLC/MAC header and a number of RLC blocks. IR operations between MS and network are performed on a RLC block basis. IR controlling (IR memory management) forms part of RLC/MAC whereas IR processing (combination of redundancy versions and subsequent decoding) is a pure PHY operation. A MS PHY receives a radio block and makes an attempt to decode the RLC/MAC header and RLC blocks. Depending on the radio conditions decoding may fail. In RLC acknowledged mode, the receiving RLC/MAC layer stores the encoded RLC block's soft values and automatically requests a retransmission by means of an ACK/NACK message sent to the network. The network side RLC/MAC layer makes sure that the network PHY uses a different puncturing pattern for the retransmission. The MS RLC/MAC layer knows when the retransmitted RLC block is received by analyzing the RLC/MAC header of each received radio block. It assures that the stored RLC block gets combined with the retransmitted version before decoding. In this manner, more redundancy bits are available at the MS after each retransmission. This procedure can be repeated various times until decoding succeeds [2].

A GSM/EDGE packet switched connection is referred to as Temporary Block Flow (TBF). Each TBF corresponds to a unique Temporary Flow Identity (TFI). RLC blocks within a TBF are numbered using the so called Block Sequence Number (BSN). Both TFI and BSN form part of the RLC/MAC header. A flow control mechanism is used in terms of a transmit window on a RLC block granularity.

A. IR in Classical GSM/EDGE Architectures

A GSM/EDGE MS architecture is typically based on two processing units and a RF transceiver. In Fig. 1 the architecture of [4] is given, comprising a DSP for baseband processing and a system processor for protocol layer handling.

The system processor has external components attached to it such as RAM and peripherals. The DSP uses accelerator blocks for computational intensive operations such as channel equalization or decoding. Alternatively, it is possible to employ dedicated hardware only for the PHY such as in [3].

IR operations are logically spread over the RLC/MAC and PHY as RLC/MAC is responsible for controlling and the PHY for rate adaption and decoding. Physically, they are spread over a DSP, an instruction set processor (system processor),

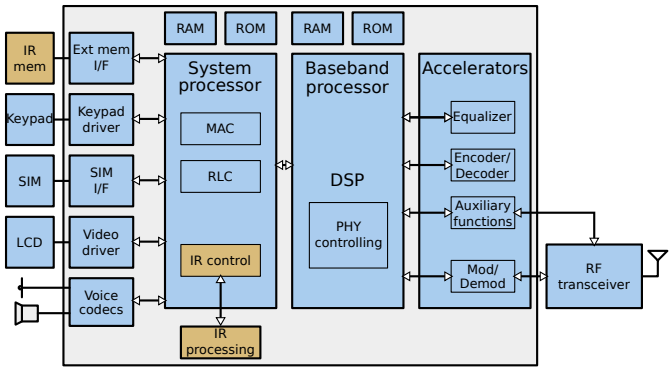

Fig. 1. Block diagram with signal processing components of a MS. IR processing as in [4] is colored in yellow.

and external RAM for storing soft value information. A large amount of data moves between PHY (DSP), RLC/MAC (instruction set processor), and IR memory (external RAM) are the result.

B. Accelerating IR Operations by Dedicated HW Architectures

In [4] an attempt was made to minimize aforementioned data moves by relieving the RLC/MAC layer from IR controlling operations. All IR controlling and processing, including decoding of RLC/MAC header and RLC blocks, were moved to a separate IR controlling and a IR processing module, respectively. They run partly on the CPU as well as in dedicated hardware as depicted in Fig. 1 in yellow. The RLC/MAC layer still needs to know whether decoding of a RLC block failed in order to produce corresponding ACK/NACK messages for the network. However, awareness of IR operations on a RLC/MAC layer level is no longer required as long as the IR controlling module is informed whether the TBF is operating in acknowledged mode. Still, a lot of data moves between IR memory (external RAM), IR controlling and processing modules (CPU and dedicated hardware), and the PHY (where equalized soft values for IR come from) remain.

Baseband processing in dedicated hardware allows further reduction of IR processing overhead, as shown in this work. We focus on PHYs as presented in [3] which allow minimization of data moves between various parts of a GSM/EDGE receiver with respect to IR. Moreover the IR complexity is completely invisible for the system processor.

However, moving IR controlling and processing into the baseband processing inherently requires some RLC/MAC layer related information to be present in the PHY. A list of TFIs corresponding to TBFs in acknowledged mode and their corresponding transmit window sizes are required. Everything else, such as BSN and puncturing version can be extracted from the decoded header bits. The decoded header is present in the PHY anyway provided that the header is actually decoded in the PHY instead of in a IR processing module attached to the system processor. Naturally, the transmit window size could simply be replaced with a hard coded value corresponding to the maximum transmit window size for

the multislot class (maximal timeslot configuration) supported (see [6]). Technically, status information for each BSN within the current transmit window per TBF is required for IR controlling to avoid decoding of already successfully received RLC blocks. In fact, it should be sufficient that IR controlling simply tracks the transmit window and clears IR memory entries corresponding to RLC blocks which drop out of the transmit window.

All things considered, it is possible and feasible to implement IR controlling and processing in the PHY by duplicating only a minimal subset of RLC/MAC layer related information. If, in addition, dedicated on-chip memory for storing soft value information is incorporated data moves with respect to IR can be reduced to

- configuration data from the RLC/MAC layer towards the PHY containing TFI information and
- notifications of decoding results from the PHY towards the RLC/MAC layer.

All other data moves, such as feeding soft values into the IR processing unit or storing soft value information in the IR memory, are inherently inevitable but can be implemented as local moves (within the PHY) only.

Variant	As in [4]	Dedicated HW (PHY)
Across layers	s_p	0
Off-chip memory read	$s_\mathrm{p} n_\mathrm{v}(s_\mathrm{d})$	0
Off-chip memory write	$s_\mathrm{p}(s_\mathrm{d})$	0
On-chip memory read	0	$s_\mathrm{p} n_\mathrm{v}(s_\mathrm{d})$
On-chip memory write	0	$s_\mathrm{p}(s_\mathrm{d})$

TABLE I
DATA MOVES FOR IR OPERATIONS.

Data moves of soft values across layers as well as corresponding IR memory accesses are summarized in Table I. The combination of a retransmitted RLC block with s_p punctured soft values with n_v redundancy versions stored in IR memory is evaluated. The depunctured RLC block consists of s_d soft values. It is assumed that only punctured redundancy versions are stored. Alternatively, whenever $s_\mathrm{p} n_\mathrm{v}$ is larger than s_d, it is wise to store depunctured RLC blocks. In that case there are s_d read and write memory accesses, irrespective of the implementation variant. Moves of control data, be it across layers or inside a layer, being negligibly small, are not considered. As Table I shows, using a dedicated hardware inside the PHY avoids cross-layer data moves. Memory accesses, however, cannot be avoided. Nevertheless, accessing an on-chip memory is more efficient as no external components are required. Keeping this savings and simplifications in mind a dedicated IR implementation inside the PHY is developed starting from an IR extension for the MatPHY framework.

III. MATPHY EXTENSIONS TOWARDS EVOLVED EDGE

The open source project MatPHY presented in [5] uses OsmocomBB (an open source GSM protocol stack [7]) and a custom PHY called *phydev* written in Matlab (see Fig. 2). The OsmocomBB distinct L1CTL protocol is used for data exchange between PHY and higher layers. MatPHY models digital baseband operations and has been structured into PHY controller, auxiliaries and three large signal processing blocks which there are Digital Front-End (DFE), Detector (DET), and Decoder (DEC). The signal processing blocks consist of controllers and various signal processing primitives.

Following the EDGE evolution MatPHY has been expanded into three directions:

- Support for packet switched data (GPRS, EDGE, Evolved EDGE Level A).
- Enhancement for the OsmocomBB L1CTL protocol.
- Configurable data source for on-the-fly IQ samples generation.

Each expansion is briefly explicated in the following.

A. Packet Switched Data Support

Packet switched data support in MatPHY is a two-fold endeavor. First, as OsmocomBB software does not support packet switched communication, a minimalistic RLC/MAC test-software is required. The latter can be used to test MatPHY packet switched operations and supports as much as receiving packet switched traffic channels. The minimalistic RLC/MAC software can communicate with the configurable data source in order to force retransmissions of RLC blocks.

Second, the phydev part of MatPHY has been enhanced for packet switched data. This includes additional controller functionality inside the various controllers including handling of packed switched L1CTL enhancements. In addition, the existing primitives for DFE, DET, and DEC were enhanced accordingly. Most notable is the required support for additional modulation schemes in DET and the incorporation of a turbo decoder for Evolved EDGE operation in DEC similar to [3]. Naturally, so far unused primitives in DEC such as depuncturing and straight forward IR with infinite memory needed to be added. Packet switched operation requires blind detection of the modulation order [2]. However, this task was omitted and a priori knowledge of the modulation order in the receiver was assumed.

B. L1CTL Enhancement

Even though the OsmocomBB project does not (yet) support packet switched data the idea of configuring the PHY by means of simple L1CTL messages seems worthwhile. Bearing this in mind, we propose an extension of the L1CTL messages for packet switched operation. In fact, only two additional L1CTL request messages are required:

`TBF_REQ`: This message holds a list of TFIs corresponding to TBFs for which the MS is currently configured. This is necessary as resource sharing is possible on the downlink [8]. The MS may distinguish payloads intended for decoding by analyzing the TFI field in the decoded header. In addition, this message indicates whether a TBF corresponding to a certain TFI is in RLC acknowledged mode or, in other words, whether IR operations are required. As described in Section II-B, the transmit window size is not indispensable for IR operations. However, if required, the configured transmit window size for

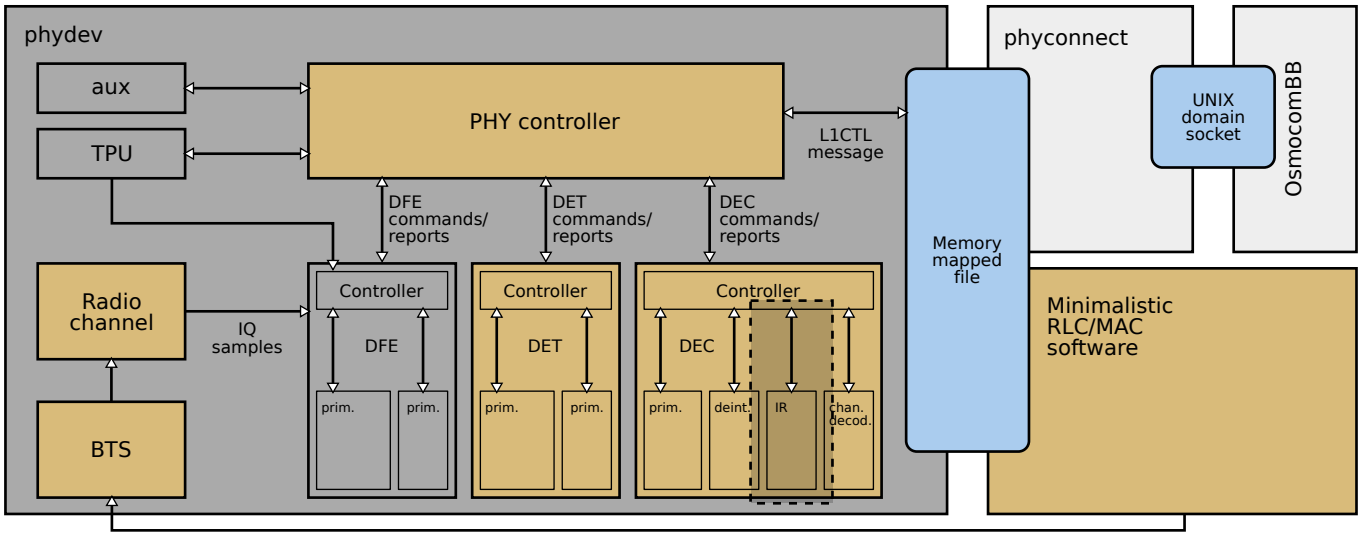

Fig. 2. MatPHY Evolved EDGE extensions: Modified or added parts as opposed to the original MatPHY are depicted in yellow. A zoom of the shaded selection in DEC can be seen in Fig. 5.

each active TBF in RLC acknowledge mode may form part of the message.

`DATA_CONF_REQ`: This message holds an information struct for each timeslot containing the Training Sequence Code (TSC) and the GSM mode (GPRS, EDGE or Evolved EDGE). In addition, it indicates whether in a certain timeslot transmission or reception is required. Furthermore, a starting frame number (FN) indicating the first valid frame of the new configuration forms part of this message. Naturally, this message can be enhanced with additional information as the need arises.

In the other direction (from PHY towards RLC/MAC) two additional L1CTL messages were defined:

`TBF_CONF`: This is a simple confirmation of the corresponding request.

`P_TRAFFIC_IND`: This message is used to indicate and report data on packet switched channels. It can be used to report RLC/MAC headers, RLC blocks and other information individually, as soon as it is decoded or, alternatively, complete radio blocks.

C. Configurable Data Source

The data source consists of a Base Transceiver Station (BTS) and a radio channel for on-the-fly IQ samples generation. The BTS can produce burst wise IQ samples for decoding. The radio channel block, on the other hand, is able to modify the transmit burst according to 3GPP channel profiles. The data source is closely connected to the minimalistic MS RLC/MAC software for e.g. the management of RLC block retransmission.

IV. Performance Evaluation

IR performance was assessed using the MatPHY extension from the previous section. As mentioned previously, soft value information of RLC blocks which could not be decoded successfully needs to be stored in the receiver. The required memory capacity is not given in the 3GPP specifications. However, a long-term throughput between the RLC/MAC and the layer on top per timeslot needs to be achieved [1].

	EDGE	Evolved EDGE
Required throughput	20 kbit/s/timeslot	33 kbit/ps/timeslot
Propagation conditions	Static, input level -97 dBm	Static, input level -94 dBm
Modulation and Coding Scheme	MCS9	DAS12
Acknowledgments polling period	32 RLC data blocks	32 RLC data blocks
Roundtrip time	120 ms	120 ms
Number of timeslots	Maximum capability of the MS	Maximum capability of the MS
Transmit window size	Maximum for the MS capability	Maximum for the MS capability

TABLE II
IR test case description [1].

Table II summarizes the test conditions under which the overall IR performance needs to be evaluated. Values such as acknowledgment polling period and transmit window size and their impact on IR test cases are described in [2]. For these simulations, the overall receiver needs to be taken into account. This includes the Noise Figure (NF) of the RF part, as well. Soft values with finite width w bits have been used in the DET and DEC blocks.

IR performance simulations have been split into two parts, one that uses parallelization and a sequential one. The extended MatPHY framework from the previous section was used in a single timeslot mode where retransmissions of RLC blocks occur immediately after decoding failed at the receiver. Correspondingly, the amount of necessary retransmissions for the IR test cases from Table II under the assumption of infinite IR memory and 0 ms roundtrip time can be recorded.

Unfortunately, such a simulation is very time consuming. Therefore, many simulations of this kind can be run in parallel in order to get results much faster and, in addition, get more accurate retransmission count information due to averaging over a larger amount of RLC blocks.

As the IR test scenario uses a static channel model, it is feasible to use the retransmission counts from the simulation to compute various decoding success probabilities of RLC blocks. There exist not more than three redundancy versions per RLC block. Therefore, the decoding success probabilities p_i for $i = 0, 1, 2, 3$ denoting the amount of redundancy versions already stored in the IR memory were computed.

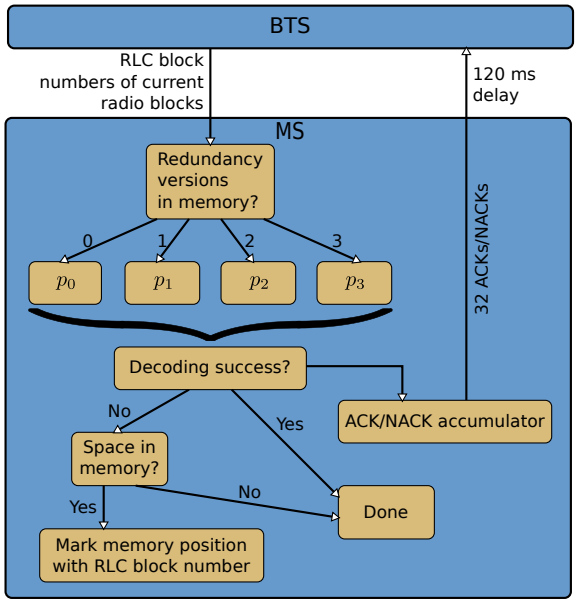

Fig. 3. IR flow setup used in the simulation.

A separate RLC block level simulation flow implementing a BTS and a MS was set up, see Fig. 3. It comprises the various requirements from Table II. In addition, the MS can be configured to various IR memory sizes in terms of punctured RLC blocks. The BTS decides using RLC/MAC procedures which RLC blocks need to be transmitted. The MS evaluates whether the decoding was successful using the decoding success probabilities described above, depending on the amount of redundancy versions already stored in IR memory. For various IR memory sizes the average throughput per timeslot was computed. Six downlink timeslots were used according to multislot class 45 as specified in [6]. As no IQ data is equalized or decoded in the RLC block level simulation, computation time is less critical and parallelization is not required.

Fig. 4 shows simulation results for MCS9 and DAS12, respectively. As required by the 3GPP IR performance requirements the throughput in kbit/s/timeslot is plotted on the vertical axis against the IR memory capacity in terms of punctured RLC blocks. Various receiver performances have been simulated. In fact, the RF performance model in terms of NF was altered whereas the digital baseband performs the same. The target throughput for MCS9 with a NF of 6 dB can be met with almost no IR memory. However, if a NF of 8 dB is used roughly 40 punctured RLC blocks need to stored. A similar increase of IR memory capacity can be observed using the same two NFs and the Evolved EDGE DAS12 test case. This clearly shows that in order to evaluate IR performance and to find a suitable IR memory size the entire PHY processing chain needs to be taken into account, including the RF transceiver. The overall receiver performance has a huge impact on IR performance.

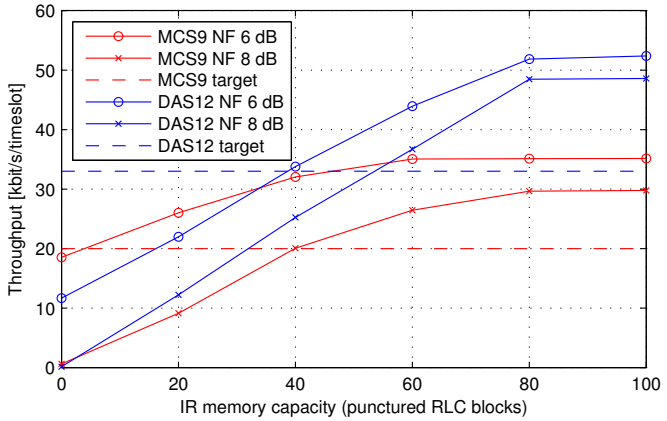

Fig. 4. Simulation results in terms of throughput (kbit/s/timeslot) against punctured RLC block memory capacity.

V. Dedicated Hardware Architecture

In order to check the feasibility of having IR operations in PHY alone a dedicated hardware architecture was developed.

A. Architecture

An enlarged version of the shaded section of Fig. 2 is depicted in Fig. 5. It consists of an IR control unit as part of the DEC controller and an IR processing unit, the latter a signal processing primitive between deinterleaver and channel decoder.

The decoded header bits, which contain among other things the BSNs of the payload, are present in the DEC controller. Consequently, it makes sense to have an IR control unit inside the DEC controller accessing the decoded header bits. All IR operations become thus completely transparent to RLC/MAC processes. The IR control unit has a small memory containing all necessary information regarding the stored RLC blocks in the IR memory such as puncturing scheme, BSN, TFI, position in IR memory, and the current transmit window position. Whenever a RLC block is ready to be deinterleaved, the IR control unit checks whether there already exists a version with the same BSN and transfers the required information to the IR processing unit. In addition, if a newly received RLC block moves the transmit window such that older RLC blocks drop out, the corresponding IR memory can be recycled. Once the decoding terminates, the IR control unit updates the control memory according to the result from the channel decoder.

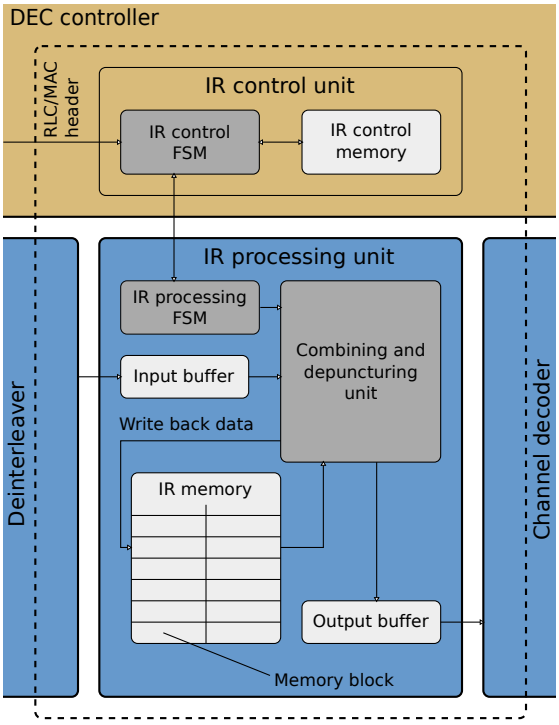

Fig. 5. Block diagram of the proposed IR architecture with dedicated memories. This corresponds to the shaded area inside DEC of Fig. 2.

The IR processing unit is responsible for depuncturing, combining previous transmissions, and storing soft values in the IR memory. Furthermore, it contains an input buffer for the newly received and punctured RLC block and an output buffer for the combined and depunctured RLC block ready to be decoded. The input buffer is used in order to write back the newly received RLC block to the IR memory in case decoding should fail. However, newly received RLC blocks are written back to the IR memory without waiting for the result of the channel decoder or, in other words, as soon as the combined version is in the output buffer. Correspondingly, the IR units can perform write back operations and process the next RLC block while the channel decoder is processing the previous RLC block. In case the same punctured version is already present in the IR memory the two transmissions are combined before write back in order to save IR memory. In order to ease memory access and management, IR memory is fragmented into memory blocks. A punctured RLC block occupies 1 or more such memory blocks depending on its size.

B. Synthesis and Clocking

The architecture as described above was implemented in VHDL and subsequently synthesized using Synopsys Design Compiler software with a 130 nm technology.

The input buffer must meet the largest punctured RLC block which consists of $s_{\mathrm{p,max}} = 1248$ soft values whereas the output buffer must be able to store the largest depunctured RLC block which consists of $s_{\mathrm{d,max}} = 2022$ soft values [9]. An example soft value width of $w = 5$ bits and an example memory capacity of $2^{15} = 32768$ soft values was used. The latter corresponds to 46 punctured or 16 depunctured DAS12 RLC blocks and in terms of MCS9 this corresponds to 53 punctured or 17 depunctured RLC blocks [9]. As can be seen in the performance simulations from the previous section MCS9 always requires less memory than DAS12 in terms of punctured RLC blocks. The memory sizes of input buffer, output buffer, IR memory, and control memory are listed in Table III. The circuit can be clocked with a maximum clock frequency of $f_{\mathrm{c,max}} = 187$ MHz which is sufficient for the fastest modes of Evolved EDGE. The circuit (without memory) corresponds to 50k Gate Equivalents (GE) at $f_{\mathrm{c,max}}$.

Memory	Soft values	Size (kbit)
Control memory	N/A	2.438
Input buffer	1248	6.24
Output buffer	2022	10.11
IR memory	32768	163.84
Total memory	36038 (without control)	182.628

TABLE III
MEMORIES AND THEIR SIZES FOR THE IMPLEMENTED ARCHITECTURE.

VI. Conclusions

The open source MatPHY framework has been extended with packet switched operation including Evolved EDGE and L1CTL extensions to connect with RLC/MAC software. IR performance with a number of receiver configurations has been studied. It has been proposed that EDGE and Evolved EDGE IR operations can be implemented efficiently as part of the PHY without burdening the system processor. RLC/MAC engineers no longer need to be aware of IR operations which simplifies protocol design. Such an implementation imposes less traffic between processors in a classical GSM/EDGE architecture. In fact, IR operations have been completely obfuscated from higher layers.

References

[1] *3GPP TS 45.005: Radio transmission and reception*, TS 45.005, Rev. 11.0.0, Jun. 2012. [Online]. Available: http://www.3gpp.org/

[2] E. Seurre, P. Savelli, and P. Pietri, *EDGE for Mobile Internet*. Artech House Publishers, 2003.

[3] C. Benkeser, A. Bubenhofer, and Q. Huang, "A 4.5 mW Digital Baseband Receiver for Level-A Evolved EDGE," in *Solid-State Circuits Conference Digest of Technical Papers (ISSCC), 2010 IEEE International*. IEEE, 2010, pp. 276–277.

[4] L. Chang and Y. Wang, "EDGE Incremental Redundancy Memory Structure and Memory Management," Jul. 23 2009, uS Patent App. 12/507,835.

[5] H. Kröll, C. Benkeser, S. Zwicky, B. Weber, and Q. Huang, "Baseband Signal Processing Framework for the OsmocomBB GSM Protocol Stack," in *Proceedings of the SDR'12 WinnComm Europe, 2012 Wireless Innovation Forum*. Wireless Innovation Forum, 2012, pp. 1–6. [Online]. Available: http://code.google.com/p/matphy/

[6] *3GPP TS 44.060: General Packet Radio Service (GPRS); Mobile Station (MS) - Base Station System (BSS) interface; Radio Link Control / Medium Access Control (RLC/MAC) protocol*, TS 44.060, Rev. 11.0.0, Mar. 2012. [Online]. Available: http://www.3gpp.org/

[7] "OsmocomBB," Jan. 2013. [Online]. Available: bb.osmocom.org

[8] *3GPP TS 43.064 General Packet Radio Service (GPRS); Overall description of the GPRS radio interface; Stage 2*, TS 43.064, Rev. 11.0.0, Sep. 2012. [Online]. Available: http://www.3gpp.org/

[9] *3GPP TS 45.003 Channel coding*, TS 45.003, Rev. 11.0.0, Sep. 2012. [Online]. Available: http://www.3gpp.org/