Adv. Radio Sci., 11, 137–142, 2013
www.adv-radio-sci.net/11/137/2013/
doi:10.5194/ars-11-137-2013

Advances in Radio Science

A code-aided synchronization IP core for iterative channel decoders

I. Ali, U. Wasenmüller, and N. Wehn

Microelectronic Systems Design Research Group, University of Kaiserslautern, 67663 Kaiserslautern, Germany

Correspondence to: I. Ali (imran@eit.uni-kl.de)

Abstract. Synchronization and channel decoding are integral parts of each receiver in wireless communication systems. The task of synchronization is the estimation of the general unknown parameters of phase, frequency and timing offset as well as correction of the received symbol sequence according to the estimated parameters. The synchronized symbol sequence serves as input for the channel decoder. Advanced channel decoders are able to operate at very low signal-to-noise ratios (SNR). For small values of SNR, the parameter estimation suffers from increased noise and impacts the communication performance. To improve the synchronization quality and thus decoder performance, the synchronizers are integrated into the iterative decoding structure. Intermediate results of the channel decoder after each iteration are used to improve the synchronization. This approach is referred to as code-aided (CA) synchronization or turbo synchronization.

A number of CA synchronization algorithms have already been published but there is no publication so far on a generic hardware implementation of the CA synchronization. Therefore we present an algorithm which can be implemented efficiently in hardware and demonstrate its communication performance. Furthermore we present a high throughput, flexible, area and power efficient code-aided synchronization IP core for various satellite communication standards. The core is synthesized for 65 nm low power CMOS technology. After placement and routing the core has an area of 0.194mm^2, throughput of 207 Msymbols/s and consumes 41.4 mW at 300 MHz clock frequency. The architecture is designed in such a way that it does not affect throughput of the system.

1 Introduction

Turbo and low-density parity-check (LDPC) codes are two most popular iterative channel codes nowadays. Due to their impressive communication performance near the Shannon limit they have become part of a wide range of wireless communication standards. As compared to non-iterative codes, they are capable of achieving a certain frame error rate (FER) at significantly reduced signal-to-noise ratios (SNR). However the performance of the iterative codes implicitly assumes ideal synchronization of the received signal. Small frequency and phase offsets result in a severe loss of communication performance in these decoders as stated in Mengali and D'Andrea (1997).

During the last decades, communication systems usually were operated at fairly high SNRs and synchronization was not a major source of communication performance degradation of a system as there are many well-established synchronization algorithms which can properly synchronize the systems at high SNR. Such synchronizers carry out estimation and correction of channel parameters prior to the decoding and function in either data-aided or non-data-aided modes. As mentioned earlier, Turbo and LDPC decoders can operate at low SNR. Consequently, the channel parameters estimation suffers from increased noise at low SNR which degrades the communication performance of the decoders. To achieve sufficient parameter estimations with these conventional synchronizers the amount of known data symbols i.e. pilot symbols had to be increased, which would result in a waste of bandwidth. Therefore in new family of synchronizers, the synchronization is integrated into the iterative decoding structure.

Integration of the synchronization into the iterative decoding process, commonly referred to as code-aided synchronization (CA Sync.), is a promising approach to synchronize a system at low SNR Herzet et al. (2007). The basic idea is to use the tentative a posteriori probability information of the transmitted bits which is available after each decoding iteration step to compute so-called soft symbols. These soft symbols are used as known data symbols as in data-aided synchronization to generate new parameter estimates; these new estimates are subsequently used to correct the received signal prior to performing further decoding iterations Godtmann and Meyr (2006).

Carrier phase, frequency and symbol timing of the received signal are three primary channel parameters. For parameter estimation and decoding the received signal must be down-converted to the baseband. Due to a slight difference of the oscillators frequencies in transmitter or receiver, a frequency offset will arise. Furthermore a frequency offset can appear by the Doppler effect if the transmitter and the receiver are in motion. Origin of the phase offset is given by the unknown transmission delay from the transmitter to the receiver which results in a random phase offset in the receiver. This paper concentrates on the frequency and phase offset estimation and correction (carrier synchronization) for turbo decoding.

At present iterative codes are part of various satellite communication standards like Digital Video Broadcasting – Return Channel via Satellite (DVB-RCS), Digital Video Broadcasting – Satellite services to Handhelds (DVB-SH) and GEO-Mobile Radio Interface (GMR-1 3G). These standards propose best combination of evolved data processing schemes at the transmitter to protect data from destructive channel effects under different transmission conditions and to support high data rate communication. While translating these requirements on the physical layer of a radio terminal, a flexible and high throughput hardware platform is highly desirable. The same hardware architecture should be able to be configured to the required air interface and also be able to accommodate future modifications of the standards.

1.1 State of the Art

A number of code-aided synchronization algorithms based on Expectation-Maximization (EM), the Newton-Raphson (NR), the Steepest-Descent (SD), the factor-graph representation and the sum-product for implementation of EM, NR and SD, pseudo-maximum-likelihood, and the correlation are proposed by different authors Godtmann and Meyr (2006), Herzet et al. (2007), Noels et al. (2005), Lottici and Luise (2004) and Wasenmüller et al. (2010). But no generic and flexible hardware implementation of the code-aided synchronization is published so far according to the best of our knowledge. This motivates us to introduce an algorithm which can be implemented efficiently in hardware and furthermore design a generic, high throughput, and flexible architecture for the code-aided synchronization to accommodate multiple wireless communication standards.

1.2 New contributions

This paper has three main contributions: (1) we demonstrate communication performance of a code-aided synchronization algorithm under different transmission parameters (2) we investigate the impact on throughput and communication performance by running the decoder and the code-aided synchronization algorithm in serial and parallel (3) we implement a CMOS synthesis based code-aided synchronization IP core to accomplish throughput, flexibility and energy efficiency requirements of various satellite communication standards. To the best of our knowledge we are the first to publish such an IP core.

This paper is structured as follows. Section 2 introduces turbo codes while Sect. 3 explains code-aided synchronization principle. In order to illustrate the advantages in communication performance due to the code-aided synchronization, simulation results are presented in Sect. 4. The hardware architecture and its implementation results are described in Sect. 5 and conclusions of this work are given in Sect. 6.

2 Turbo codes

With the introduction of binary turbo codes by Berrou in 1993 Berrou et al. (1993) near optimum error correction became possible. Due to these error correction capabilities, binary and duo-binary turbo codes allow for low frame error rates (FER) at a low signal-to-noise ratio (SNR). Turbo codes generally consist of a serial or parallel concatenation of two codes, so called component codes, and an interleaver.

Decoding of turbo codes is an iterative process where probabilistic information is exchanged between component decoders. A possible realization of a decoder for turbo codes is given in Fig. 1. The two component decoders that decode the two component codes are connected via interleaver and deinterleaver. They use *log likelihood ratio*s (LLR) $\lambda^{s}_{d_k}$, $\lambda^{p^1}_{d_k}$ and $\lambda^{p^2}_{d_k}$ of the systematic and parity information to compute the extrinsic information $\Lambda^{e1}_{d_k}$ and $\Lambda^{e2}_{d_k}$ on the information bits. The iterative exchange of $\Lambda^{e1}_{d_k}$ and $\Lambda^{e2}_{d_k}$ between these component decoders is referred to as turbo principle. One (full) iteration is done if Decoder 1 and Decoder 2 have run once. If only one decoder has calculated new information, we call this one half iteration.

The decoding algorithm consists of a forward and a backward recursion. It computes for each possible information or parity bit d_k an a posteriori probability (APP) LLR: $\Lambda^{s}, \Lambda^{p1}, \Lambda^{p2}$.

3 Code-aided synchronization

The synchronization consists of the estimation of the unknown parameters of timing, frequency and phase offset, and the elimination of all possible negative influences introduced by these parameters. We focus on the frequency and phase synchronization in conjunction with turbo decoding. We assume, that the steps of gain control, timing synchronization, frame detection and coarse synchronization are properly carried out before. The received sample sequence r is given in the complex baseband according to Eq. (1):

$$r(l) = s(l) \cdot e^{j(2\pi f_o l + \Phi)} + n(l) \qquad l = 0, 1, \ldots, L-1 \qquad (1)$$

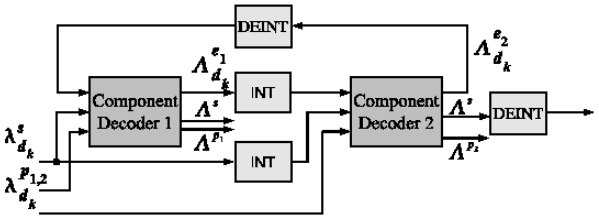

Fig. 1. Turbo Decoder.

The sample sequence r with L elements is based on modulation symbols $s(l)$ with one sample per symbol and symbol duration T, and is disturbed by a noise sequence n.

The unknown parameters of frequency offset f_o and phase offset Φ have to be estimated for every received sample sequence. These parameters are considered fixed during an estimation interval; the sample sequence has to be corrected accordingly to the estimated frequency offset $\tilde{f}_o$ and estimated phase offset $\tilde{\Phi}$. The synchronization is done in two main steps. Initially, a (coarse) carrier synchronization is performed with the help of pilot symbols. Afterwards fine synchronization is done iteratively with the additional use of tentative decoder decisions after each decoder iteration which is called here code-aided (CA) synchronization. This principle improves the estimation parameters for synchronization. The improved carrier synchronization is used to provide better input data for the decoding process. This process is done iteratively after each decoder iteration. Frequency and phase offset can be optimally estimated on an unmodulated carrier. With the assumption, that the transmitted symbol sequence s of the burst is known the effect of the modulation by each transmitted symbol $s(l)$ can be removed by:

$$\tilde{r}(l) = r(l) \cdot s^*(l) \qquad l = 0, 1, \ldots, L-1 \tag{2}$$

However it must be considered, that usually the symbols of the burst are unknown or only some symbols, used for supporting the burst detection or supporting the coarse synchronization are known. Thus we replace the transmitted symbol sequence s by an estimated symbol sequence s_e. The estimation of the transmitted symbol sequence is provided by the turbo decoder. The code-aided estimation of frequency and phase offset is based on the average phase $\tilde{Z}_0$ of the front part and on the average phase $\tilde{Z}_1$ of the rear part of the burst with a modulation removal by the estimated symbol sequence s_e. This is formally given by:

$$\tilde{Z}_k = \sum_{l=0+k\cdot L/2}^{L/2-1+k\cdot L/2} r(l) \cdot s_e^*(l) \qquad k = 0, 1 \tag{3}$$

With the two phase values of Eq. (3) the estimate of the frequency offset can be calculated with:

$$\tilde{f}_0 = \frac{\arg(\tilde{Z}_0 \cdot \tilde{Z}_1^*)}{2\pi \cdot L} \tag{4}$$

The estimate of the phase offset is calculated with the help of Eq. (4):

$$\tilde{\phi} = \arg(\tilde{Z}_0 + \tilde{Z}_1) - L \cdot \tilde{f}_0 \cdot \pi \tag{5}$$

The first decoder iteration is based on the LLR values $\lambda_{d_k}^s$, $\lambda_{d_k}^{p^1}$ and $\lambda_{d_k}^{p^2}$ calculated with the symbols of the coarse synchronized received sequence. For the code-aided synchronization an estimate of the transmitted symbols is used, which is produced by the turbo code decoder after each iteration. The estimate of the transmitted symbols is gathered by the APP LLR of the decoder. A turbo code decoder computes APP LLR values Λ^s of the systematic bits by default. In code-aided synchronization applications the decoder must additionally calculate the APP LLR values $\Lambda^{p1,2}$ for the parity bits. With the tentative soft values of systematic bits and parity bits the sequence s_e is generated and the described code-aided synchronization process can be carried out. The LLR are converted into symbols according to the Eq. (6):

$$s_e(l) = \tanh(\frac{\tau^x(2l)}{2}) + j\tanh(\frac{\tau^x(2l+1)}{2}) \qquad l = 0, 1, \ldots, L-1 \tag{6}$$

Here τ^x respresents the sequence of τ^s, τ^{p1} and τ^{p2}, which corresponds to the transmitted symbols sequence r. The received sequence r is corrected with the new estimates of frequency and phase offset. The symbols correction is performed according to the Eq. (7):

$$r(l) = r(l) \cdot e^{-j(2\pi \tilde{f}_o l + \tilde{\Phi})} \qquad l = 0, 1, \ldots, L-1 \tag{7}$$

In this way a new synchronized received sequence r^* is calculated after each full decoder iteration. The above presented code-aided synchronization algorithm is very similar to that presented in Luise and Reggiannini (1995) and Wasenmüller et al. (2010).

4 Communication performance simulations

In this section, we demonstrate the communication performance of our system. The simulations were carried out with bit true models of the hardware units to take into account the quantization losses. It is assumed that ideal timing and and frame synchronization are done in advance. Furthermore a coarse carrier synchronization is assumed, which resolves phase ambiguities and limits the residual phase and frequency offset. In current research only QPSK modulation scheme was adopted and no pilot symbols were used in the bursts. In all experiments, we used an additive white Gaussian noise (AWGN) channel. To demonstrate the performance of our CA-synchronization algorithm in challenging conditions, we chose a constant value of frequency offset. That results in linearly increasing/decreasing phase offset on

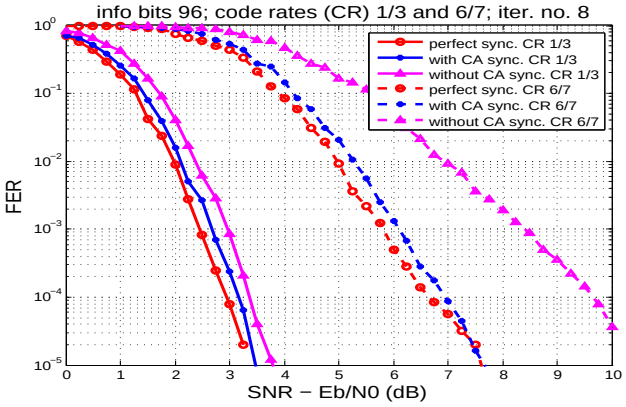

Fig. 2. Performance comparison, with and without CA Sync., short block length, in presence of residual phase and frequency offset.

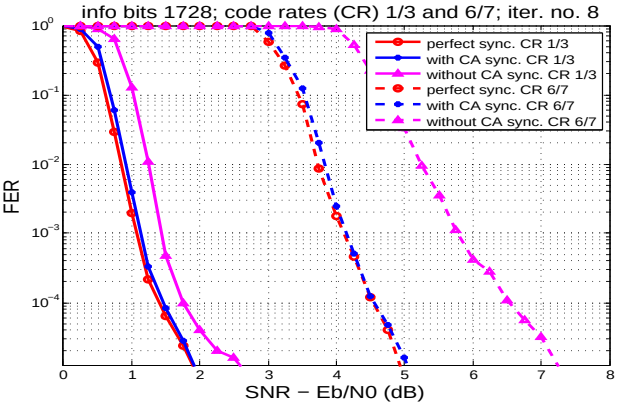

Fig. 3. Performance comparison, with and without CA Sync., long block length, in presence of residual phase and frequency offset.

each symbols of a block. So we calculated the constant frequency offset value for each block depending on its length. The first symbol of the block gets a rotation of $-22.5°$ then the offset linearly increases for each symbol in a block and the middle symbol gets 0° phase offset and the last symbol gets a phase offset of 22.5°. If the input symbols are rotated with an offset of $\pm 45°$ with QPSK modulation then the decoder can not decode the data that's why in all experiments we kept the offset within the limit ($\pm 45°$). To get each result 250 000 blocks were simulated.

We used an 8-state duo-binary turbo decoder in our simulations having Max-Log-Map with ESF (extrinsic scaling factor) algorithm having value 0.75, sliding window with one trellis stage, next iteration initialization (NII) acquisition, the window length equal to the information bits, 8 iterations, 6 bit quantization for the input LLR and 7 bit for the extrinsic LLR.

We present frame error rate (FER) graphs to show the impact on communication performance of different parameters: frequency offset, phase offset, block length and code rate. Moreover, the effect on communication performance by running the synchronization in parallel to the decoder is also shown.

4.1 Phase and frequency offsets impact

Figures 2 and 3 show the performance of the decoder in presence of residual phase and frequency offsets. FER curves with ideal synchronization i.e. zero phase and frequency offset, with and without the CA synchronization are also shown in the figures. It is evident that the residual phase and frequency offsets degrade the decoder performance but the CA synchronization significantly improves the performance, i.e. the communication performance is close to the ideal synchronization by doing CA synchronization.

4.2 Block length and code rate variations

We carried out simulations with different block lenghts and code rates in presence of residual phase and frequency offsets. In this paper, we present the communication performance of the CA synchronization with the shortest and the longest block lengths and, the lowest and the highest code rates of DVB-RCS standard. The Figs. 2 and 3 show the results with 96 and 1728 bits block lengths and, 1/3 and 6/7 code rates. The gain in communication performance due to CA synchronization depends upon code rate, block length, and residual phase and frequency offset. The figures show that at FER 10^{-3} the performance gain is almost 0.25 dB with code rate 1/3 and almost 1.6–2.0 dB with code rate 6/7 depending upon the block length.

4.3 Synchronization running in parallel impact

If the synchronization and the decoder function in serial, i.e. the synchronizer waits for completion of the decoder iteration, then latency of the system is increased. As a consequence, the system throughput is heavily degraded. On the other hand, if the synchronization and the decoder function in parallel, i.e. both start their functions at the same time, but the synchronization unit uses previous iteration results of the decoder, then in this case there is no effect on the system throughput but the communication performance gain of the CA synchronization is degraded slightly. We show communication performance graphs of the both situations in Fig. 4 which shows that we get a penalty in communication performance when the code-aided synchronization is running in parallel to the decoder: 0.1 dB at iteration number 4 and 0.05 dB at iteration number 8. The insignificant loss at higher number of the decoder iterations instigates us to design an architecture running in parallel to the decoder.

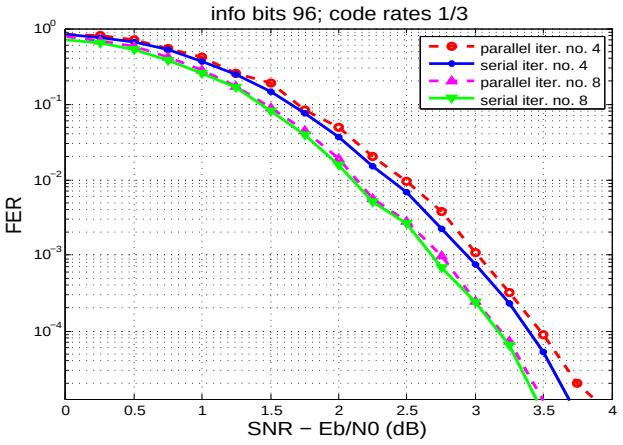

Fig. 4. Performance comparison, CA Sync. running in serial and parallel with the decoder.

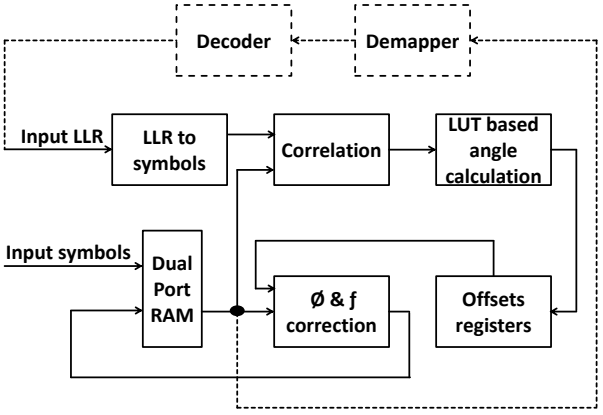

Fig. 5. Architecture of the code-aided synchronization.

5 Architecture and its implementation

The architecture of the system is shown in Fig. 5. We use synthesizable VHDL for modeling and implementing the architecture. The modules shown with dotted box in the figure, demapper and decoder, are not part of the architecture and are shown here just for architecture operation explanation. We use Lookup Tables (LUTs) to store the values of tanh, sin and cos which are required to convert the input LLR into symbols and for the symbols correction. We use STMicroelectronics ROMs for implementing all the LUTs and a DPR (dual port ram) 3072×20 to store the symbols. Two complex MAC (Multiply-Accumulate) units are implemented to perform the correlation and correction functions in parallel. Second term of Eq. (5) is also mapped on the correlation MAC unit to save an additional multiplier. Moreover a divider is implemented to get the frequency offset as described in Eq. (4).

5.1 LUT based phase calculation

We use a LUT for phase calculation (Eq. (4) and Eq. (5)) instead of using a full CORDIC (Coordinate Rotation Digital Computer). The address of the LUT is generated by the correlation function. However, the output of the correlation function is a complex number having large bitwidth. So it is required to reduce the bitwidth of the number in order to reduce the LUT size. Instead of directly truncating LSBs of a complex number which can result a number with less information carrying bits, we do scaling in a smart way. So first we truncate all possible no information carrying MSBs and then the complex number is shortened and rounded to 10 bits.

5.2 Operation

After one iteration of the turbo decoder, LLRs of the systematic and parity bits are available at the output of the decoder. Two LLR values are transferred to the "LLR to symbols" module in one clock cycle to convert them into a symbol for QPSK modulation for implementation of Eq. (6). The symbols received by the AWGN channel are stored into the DPR. One symbol from the DPR and the soft symbol at the output of the "LLR to symbols" module are transferred to the "Correlation" module. One symbol is processed in one clock cycle and in this way the whole block is processed.

The "Correlation" module implements Eq. (3) and its result is transferred to the "LUT based angle calculation" module for computation of angles (see argument function in Eq. (4) and Eq. (5)). The "Offsets registers" module saves the results of Eq. (4) and Eq. (5) for current iteration. The "Ø & f correction" module implements Eq. (7).

5.3 Synchronization running in parallel to decoder

To run the synchronizer in parallel to the decoder, the phase & frequency (Ø & f) correction module remains idle during the first decoder iteration. When the decoder executes its second iteration and the LLR are available at its output and then the correlation and correction modules start their operations in parallel. A symbol is read from the DPR and it is transferred to both the correlation and correction modules, in the same clock cycle. In this way we save a lot of accesses to the DPR. So the correlation is performed on the decoder $(n+1)$-th iteration output while the correction is performed on the decoder n-th iteration output. One symbol is corrected in one clock cycle and it is saved into the DPR again. So the DPR is used to save the corrected symbols also. After correction of a complete block, it is transferred to the demapper for next iteration of the decoder.

5.4 Implementation results

The code-aided synchronization IP Core is implemented on a 65 nm low power CMOS library. We considered the

Table 1. Post-layout area results.

Component	Case	Max. clock freq.	Area
DPR	Worst	300 MHz	0.089 mm^2
Logic	Worst	300 MHz	0.041 mm^2
ROMs (LUTs)	Worst	300 MHz	0.064 mm^2
Sum			0.194 mm^2

following process, voltage and temperature (PVT) parameters: Worst Case (WC, 1.1V, 125 °C), Nominal Case (NOM, 1.2V, 25 °C) and Best Case (BC, 1.3V, −40 °C). Synthesis was performed with Synopsys Design Compiler in topographical mode, placement and routing (P&R) with Synopses IC Compiler. The DPR can operate at maximum 300 MHz clock frequency so we synthesized the core at this frequency. Table 1 shows the results for area and the final physical design is shown in Fig. 6. We calculated the power consumption of the physical design with Synopsys PrimeTime-PX. The design consumes 51.0, 41.4 and 35.4 mW at best, nominal and worst case, respectively, at 300 MHz clock frequency.

The synthesis results show that we save a significant area by using a LUT for phase calculation. It occupies only 0.0186 mm^2 area while a complete CORDIC implemented on a 65 nm ASIC takes 0.86 mm^2 area Qi et al. (2010). However the CORDIC can also perform several other functions like conversions between polar and rectangular coordinates.

5.5 Flexibility

We analyzed requirements for utilizing the IP core for different satellite communication standards like DVB-RCS, DVB-SH, and GMR-1 3G and to support these standards we provided flexibility in the architecture at: (i) run time – block length and modulation scheme (ii) design time – bitwidths of the input symbols, the decoder LLR, and the intermediate results, and maximum number of iterations.

5.6 Throughput

The architecture has a throughput of 207 Msamples/sec for one iteration. This calculation includes both systematic and parity bits after depuncturing a block. This throughput is sufficient to meet various satellite communication standards requirements.

6 Conclusions

The code-aided synchronization improves the communication performance of the iterative decoders with a gain of 0.25 to 2.0 dB at FER 10^{-3} depending on block length, code rate, and residual phase and frequency offset. In this paper, we presented, to the best of our knowledge, the first published IP core for the code-aided synchronization. The core functions in parallel to the decoder so it does not degrade its throughput.

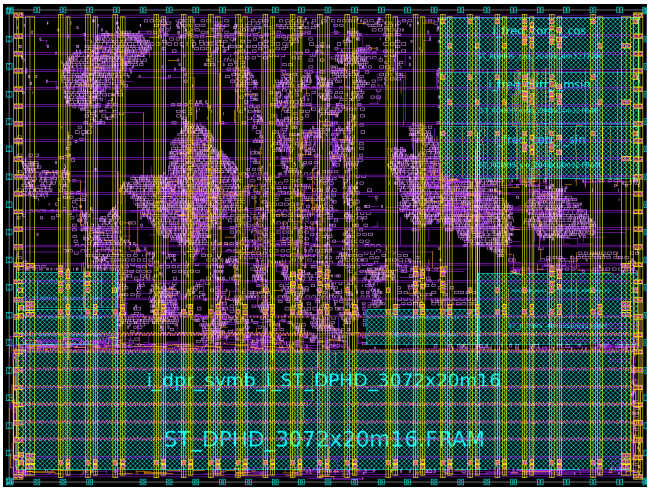

Fig. 6. Physical design including the DPR.

References

Berrou, C., Glavieux, A., and Thitimajshima, P.: Near Shannon Limit Error-Correcting Coding and Decoding: Turbo-Codes, in: Proc. 1993 International Conference on Communications (ICC '93), 1064–1070, Geneva, Switzerland, 1993.

Herzet, C., Noels, N., Lottici, V., Wymeersch, H., Luise, M., Moeneclaey, M., and Vandendorpe, L.: Code-Aided Turbo Synchronization, Proceedings of the IEEE, 95, 1255–1271, 2007.

Lottici, V. and Luise, M.: Embedding Carrier Phase Recovery Into Iterative Decoding of Turbo-Coded Linear Modulations, IEEE Transactions on Communications, 52, 661–669, 2004.

Luise, M. and Reggiannini, R.: Carrier frequency recovery in all-digital modems for burst-mode transmissions, Communications, IEEE Transactions on, 43, 1169–1178, 1995.

Mengali, U. and D'Andrea, A.: Synchronization Techniques for Digital Receivers, Plenum Publishing Corporation, New York, 1997.

Noels, N., Lottici, V., Dejonghe, A., Steendam, H., Moeneclaey, M., Luise, M., and Vandendorpe, L.: A Theoretical Framework for Soft-Information-Based Synchronization in Iterative (Turbo) Receivers, EURASIP J. Wireless Comm. and Networking, 2005, 117–129, 2005.

Qi, Z., Cabe, A., Jones, R., and Stan, M.: CORDIC implementation with parameterizable ASIC/SoC flow, in: IEEE SoutheastCon 2010 (SoutheastCon), Proceedings of the, 13–16, doi:10.1109/SECON.2010.5453930, 2010.

Godtmann, S., Hadaschik, N., Steinert, W., Pollok, A., Ascheid, G., and Meyr, H.: Coarse and Turbo Synchronization: A Case-Study for DVB-RCS, in: NEWCOM-ACoRN Workshop, Vienna, Austria, 2006.

Wasenmüller, U., Gimmler, C., and Wehn, N.: Low complexity Turbo synchronization without initial carrier synchronization, Advances in Radio Science, 8, 123–128, 2010.