BCP Usage in PDSCH Receiver

NOTE: BCP is not designed for UE or Relay node processing and Texas Instruments does not support this mode. This document is being provided purely for information purposes. Texas Instruments does not hold any responsibility related to success of the application should the user decide to use this information.

Item	Document name	Version	Description
1	BCP User guide		
3	3GPP TS-36.211	8.9.0	Physical Channels and Modulation (Release 8) http://www.3gpp.org/ftp/specs/html-info/36211.htm
4	3GPP TS-36.212	8.8.0	Multiplexing and Channel Physical Channels and Modulation (Release 8) http://www.3gpp.org/ftp/specs/html-info/36212.htm

BCP Usage in PDSCH Receiver

BCP is designed explicitly to accelerate PUSCH/PUCCH receivers and PDSCH/PDCCH transmitters, but not PDSCH receiver that exists in both UE and Relay Node (RN), in LTE applications [1]. However, the bit processing of PDSCH receiver may be offloaded to BCP partly with some work-around aids provided by CPU and EDMA.

The BCP usage details depend on granularity of the noise scaling factors. In this document, two kinds of granularities are considered:

1. One noise scaling factor per TB,
2. One noise scaling factor per resource element (RE).

One Noise Scaling Factor per TB

Figure 1 depicts the PDSCH receiver signal chain and the mapping from the signal chain components to the processing resources, when one TB has only a single noise scaling factor.

1. All the Cmux scaling factors configured in the SSL local header are set to the same common value.
2. According to section 5.1.4.1.2 of [4], the main difference between PUSCH and

PDSCH rate de-matching is the limited soft-buffer size at the receiver side for PDSCH receiving. Accordingly, the fields defined in the BUFFER_WRAP_OUT_CFG word of the RM's local header specify the wrap-around position in the P1/P2 dual matrix, but RD has no similar local header fields and can't be

used to perform rate de-matching in PDSCH receiver on UEs and RNs with limited soft-buffer capacity [1].

However, RD can be used on the UEs and RNs which are not soft-buffer limited. Note that for the KeyStone-based RNs, normally the soft-buffer should be large enough, so RD can be used to offload the rate de-matching and HARQ combining.

- To disguise PDSCH as PUSCH for using SSL, the PUSCH channel de-interleaving on complex symbols and the PUSCH channel interleaving on soft bits are performed before and after SSL processing, respectively. The former can be done by CPU on-the-fly when performing equalization. The latter can be done by EDMA.

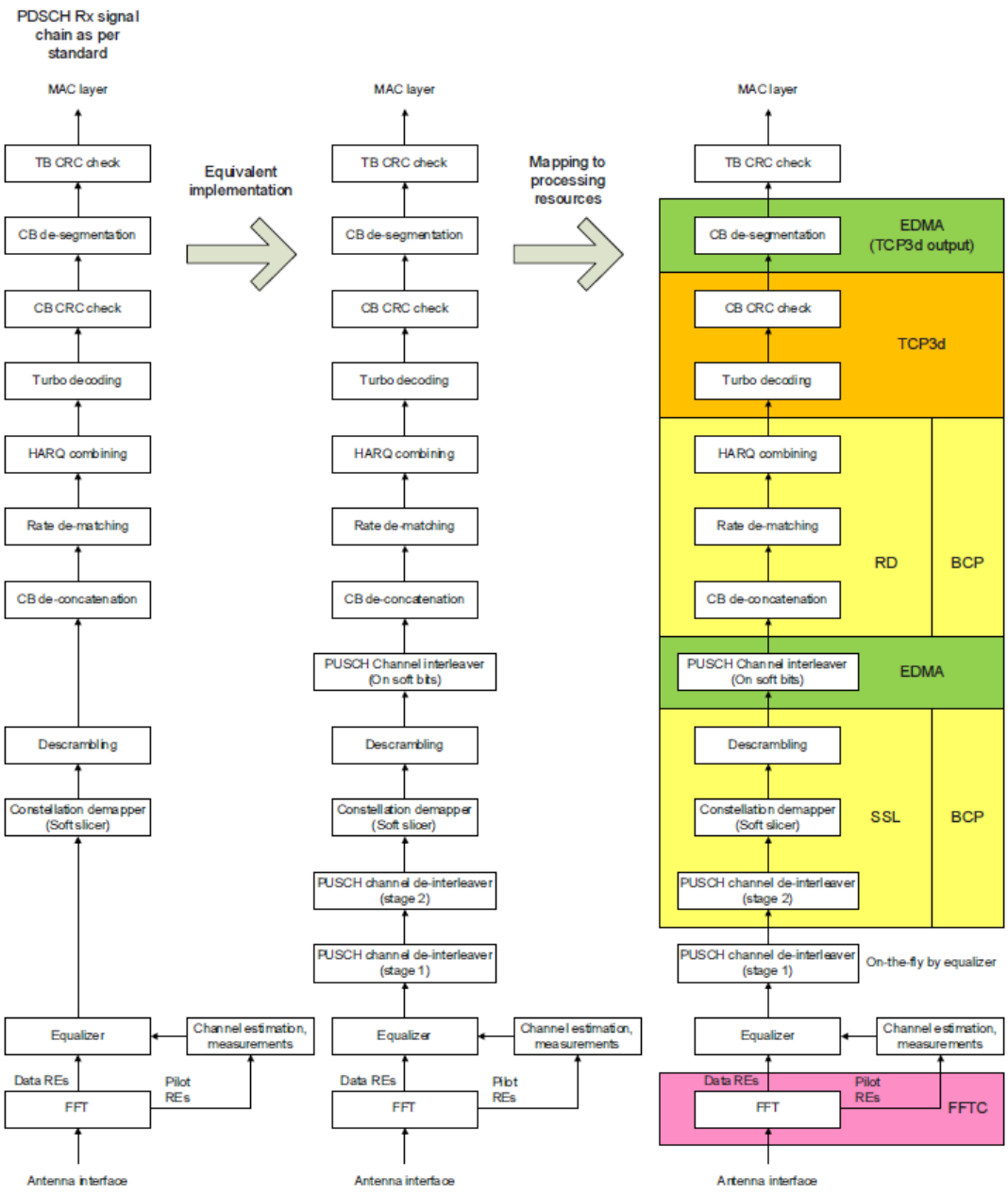

Figure 1: PDSCH receiver signal chain and processing resource mapping (one noise scaling factor per TB)

Input Data of SSL

SSL is designed to perform channel de-interleaving, soft slicing, descrambling and UCI handling for PUSCH. Soft slicing is performed symbol by symbol and doesn't depend on the data order, but channel de-interleaving and descrambling depend on the data order. To use SSL for PDSCH processing, the equalizer output data have to be format converted before sent to BCP as if we are processing PUSCH from the SSL point of view. This conversion is performed in the "PUSCH channel de-interleaver (stage 1)"

stage in Figure 1, and can be done by the equalizer on-the-fly. Figure 2 shows an example of the format conversion

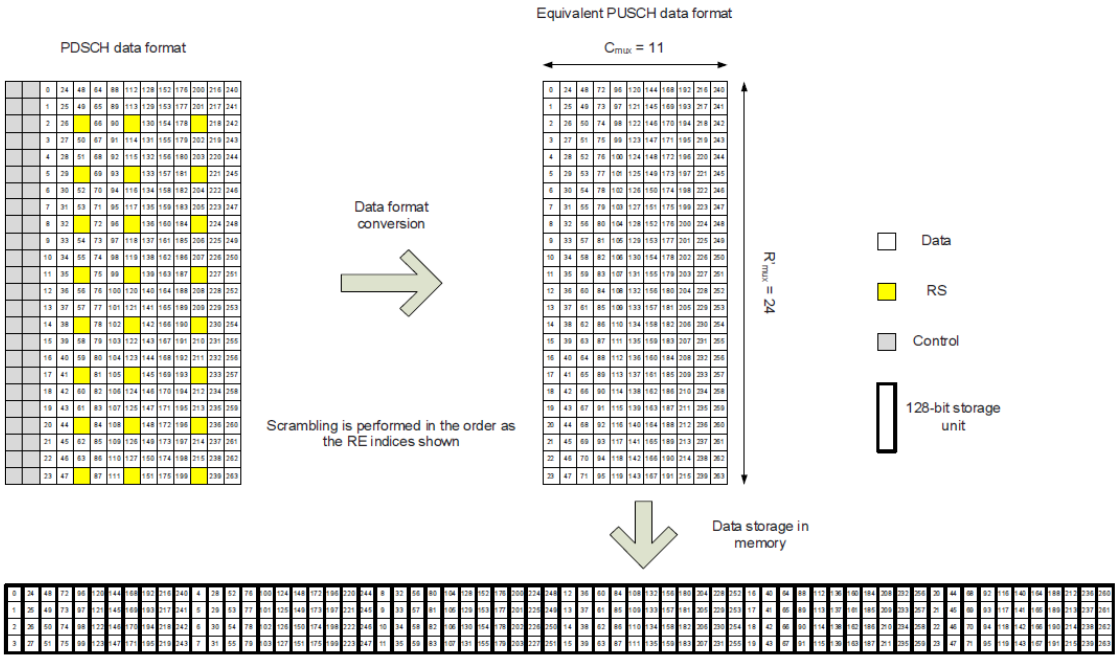

Figure 2: Format conversion of PDSCH equalized data

According to [3], PDSCH is transmitted for each involved antenna port on REs not used for PCFICH, PHICH, PDCCH, PBCH, synchronization signals or reference signals. In general, denote the number of REs on one layer as Msymb. The Rmux index (section 5.2.2.8 of [4], Figure 52 of [1]) and Cmux (section 5.2.2.8 of [4]), that have to be configured in SSL local header, can be derived from Msymb as below.

- Search for the minimum M'symb in Table 1 such that M'symb >= Msymb.
- Get Rmux index and Cmux from that line.

If the TB maps to two layers, the symbols of the two layers should be interleaved before sent to BCP as shown in Figure 40 of [1], but the way to get the Rmux index and Cmux is still the same, and the Msymb used is the number of REs of a single layer (instead of two layers). If M'symb > Msymb, (M'symb - Msymb) padding symbols per layer have to be appended at the end of the SSL input symbol sequence.

Rmux and Cmux are used not only to configure the SSL local header, but also to get the output position of each equalized symbol as shown in Figure 2.

Table 1: Mapping table from M'_{symb} to R_{mux} and C_{mux}

Index	M'_{symb} (Single layer)	R_{mux} Index	C_{mux}	Index	M'_{symb} (Single layer)	R_{mux} Index	C_{mux}
0	108	0	9	**53**	3456	14	12
1	120	0	10	**54**	3564	16	11
2	132	0	11	**55**	3600	15	12
3	144	0	12	**56**	3840	18	10
4	216	1	9	**57**	3888	16	12
5	240	1	10	**58**	3960	17	11
6	264	1	11	**59**	4224	18	11
7	288	1	12	**60**	4320	17	12
8	324	2	9	**61**	4608	18	12
9	360	2	10	**62**	4752	19	11
10	396	2	11	**63**	4800	20	10
11	432	2	12	**64**	4860	21	9
12	480	3	10	**65**	5184	19	12
13	528	3	11	**66**	5280	20	11
14	540	4	9	**67**	5400	21	10
15	576	3	12	**68**	5760	20	12
16	600	4	10	**69**	5832	24	9
17	648	5	9	**70**	5940	21	11
18	660	4	11	**71**	6000	23	10
19	720	4	12	**72**	6336	22	11
20	792	5	11	**73**	6480	21	12
21	864	5	12	**74**	6600	23	11
22	960	6	10	**75**	6912	22	12
23	972	7	9	**76**	7128	24	11
24	1056	6	11	**77**	7200	23	12
25	1080	7	10	**78**	7680	26	10
26	1152	6	12	**79**	7776	24	12
27	1188	7	11	**80**	7920	25	11
28	1200	8	10	**81**	8100	28	9
29	1296	7	12	**82**	8448	26	11

29	1296	7	12	**82**	8448	26	11
30	1320	8	11	**83**	8640	25	12
31	1440	8	12	**84**	8748	30	9
32	1584	9	11	**85**	9000	28	10
33	1620	10	9	**86**	9216	26	12
34	1728	9	12	**87**	9504	27	11
35	1800	10	10	**88**	9600	29	10
36	1920	11	10	**89**	9720	30	10
37	1944	12	9	**90**	9900	28	11
38	1980	10	11	**91**	10368	27	12
39	2112	11	11	**92**	10560	29	11
40	2160	10	12	**93**	10692	30	11
41	2304	11	12	**94**	10800	28	12
42	2376	12	11	**95**	11520	29	12
43	2400	13	10	**96**	11664	30	12
44	2592	12	12	**97**	11880	31	11
45	2640	13	11	**98**	12000	33	10
46	2700	15	9	**99**	12672	32	11
47	2880	13	12	**100**	12960	31	12
48	2916	16	9	**101**	13200	33	11
49	3000	15	10	**102**	13824	32	12
50	3168	14	11	**103**	14256	34	11
51	3240	16	10	**104**	14400	33	12
52	3300	15	11	**105**	15552	34	12

Channel Interleaving EDMA

The SSL output data have to be channel interleaved before sent to RD as shown in Figure 1. This can be performed by EDMA. Figure 3 depicts the EDMA transfer pattern for a single TB. The EDMA PaRAM set parameters can be derived from the figure.

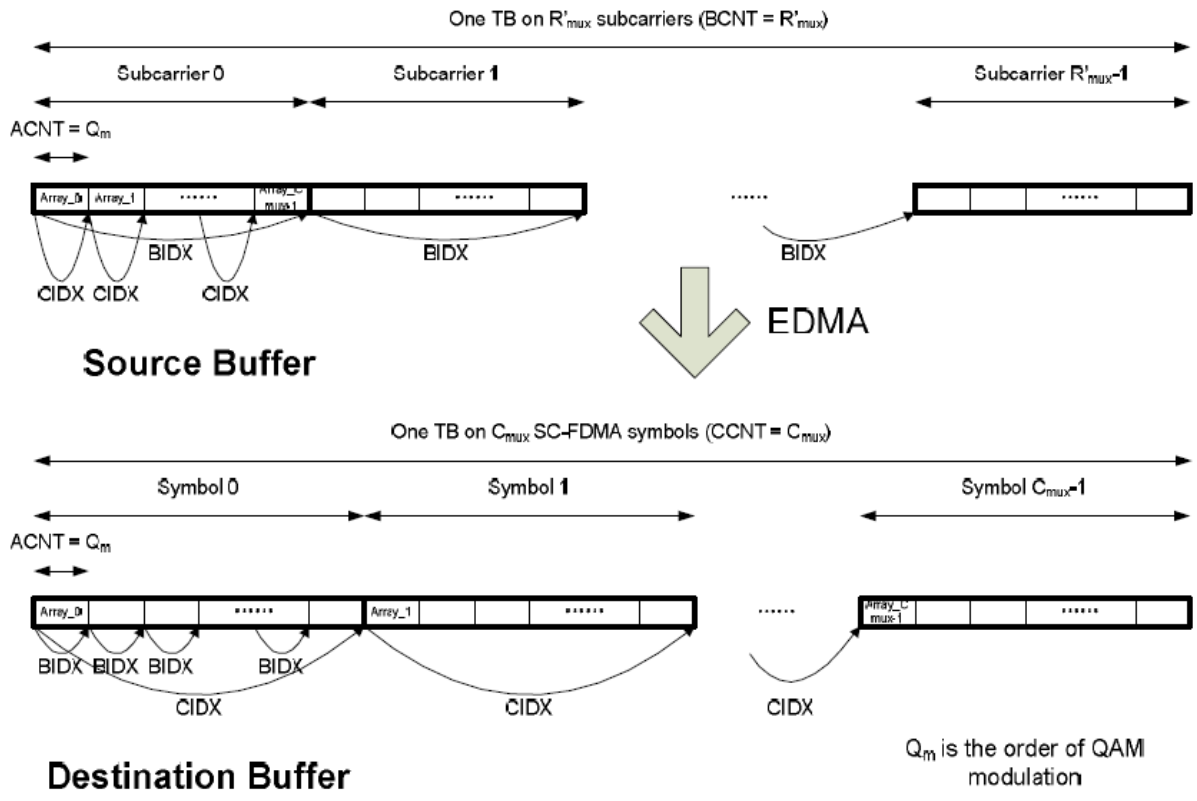

Figure 3: EDMA transfer pattern of the channel interleaving

RN can be seen as a special UE from the DeNB (Donor eNB) point of view. There is only at most one allocation scheduled for RN in one TTI, so at most two TBs (when in spatial multiplexing mode with more than two layers) or one TB (in other scenarios) is sent from DeNB to RN in one TTI. For simplicity, we don't pursue automatic connection from SSL to the channel interleaving EDMA. Instead, we can use the queue-pend queue as Rx queue to generate interrupt to notify CPU the completion of SSL processing. Then CPU can trigger the EDMA manually.

One Noise Scaling Factor per RE

Figure 5 depicts the PDSCH receiver signal chain and the mapping from the signal chain components to the processing resources, when each RE has its own noise scaling factor.

If each RE has it own noise scaling factor, we have to use WiMAX mode, instead of LTE mode, configured in the BCP global header, because only in this mode each input QAM symbol is provided with its own noise scaling factor (section 5.7.1.3 of[1]).

In WiMAX mode, SSL only supports the soft slicing function, leaving descrambling (including scrambling sequence generation) done by CPU.
To make the descrambling processing easier at the output side of SSL, the equalized symbols should be arranged in the scrambled order at the input side of SSL.
In the RN application, for simplicity we can use the queue-pend queue to generate interrupt to notify CPU the completion of the SSL processing. Then CPU performs descrambling and starts BCP (for RD).

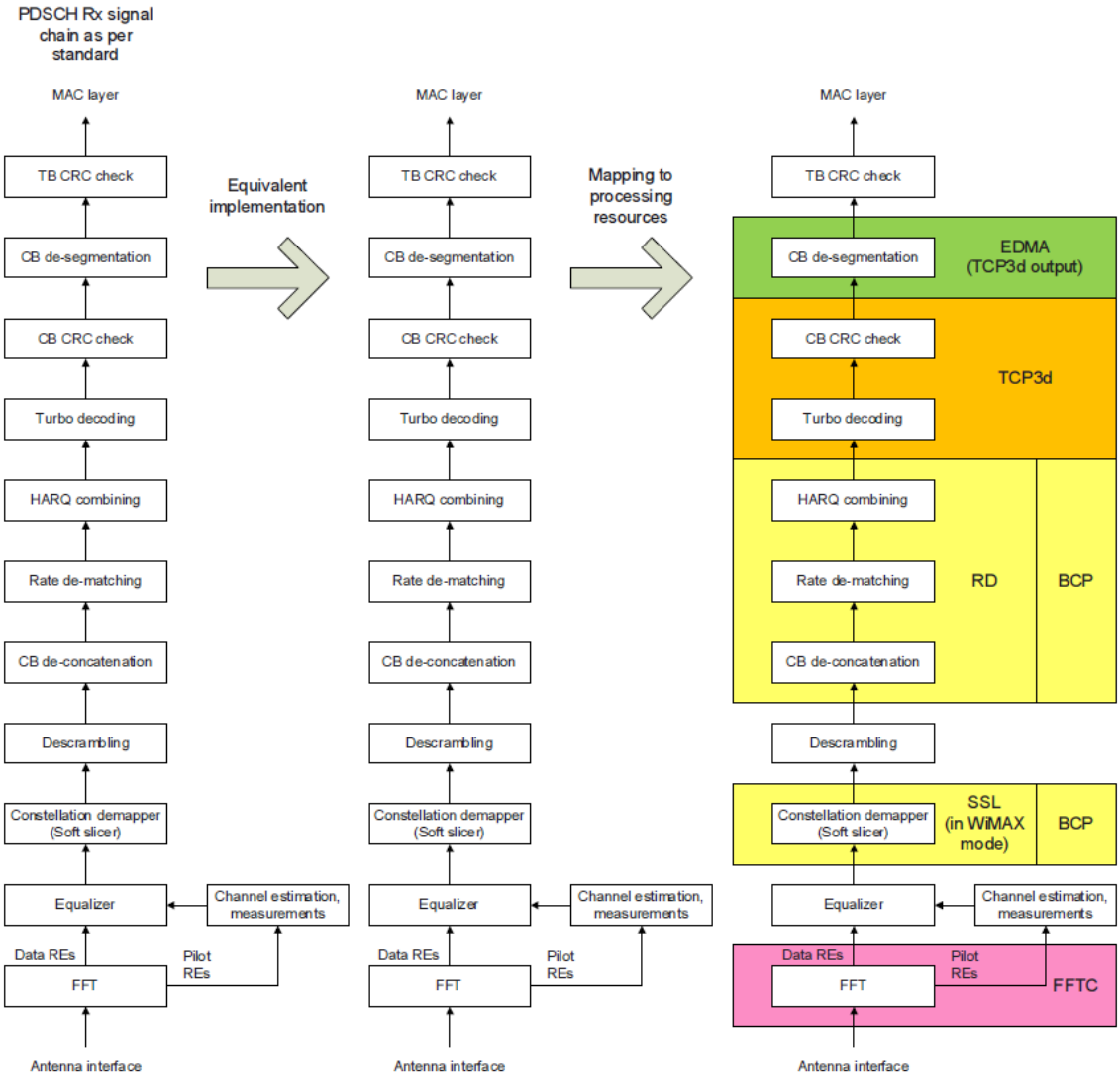

Figure 5: PDSCH receiver signal chain and processing resource mapping (one noise scaling factor per RE)