组会报告

徐益

2018 年 11 月 19 日

1 工作内容

1. Debug 连调程序并测试；
2. 阅读协议并修改资源映射结构。
3. 根据新的资源映射结构修改程序

2 Debug 连调程序并测试

2.1 Debug 结果

MAC 层接收端读参数标识早于 PHY 层接收端写参数标识
最终导致读写冲突。

2.2 HARQ 测试结果

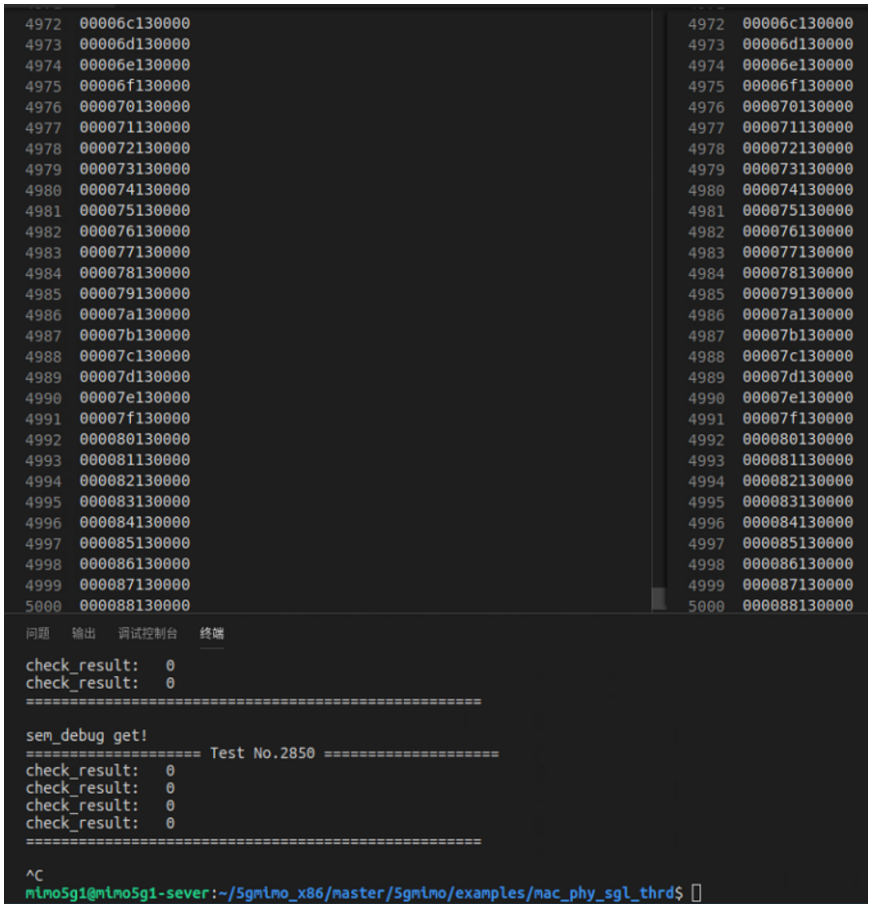

图 1: HARQ 测试结果

3 阅读协议并修改程序结构

3.1 正确的层映射关系

Number of layers	Number of codewords	Codeword-to-layer mapping $i = 0,1,\ldots,M_{\text{symb}}^{\text{layer}}-1$	
1	1	$x^{(0)}(i) = d^{(0)}(i)$	$M_{\text{symb}}^{\text{layer}} = M_{\text{symb}}^{(0)}$
2	1	$x^{(0)}(i) = d^{(0)}(2i)$ $x^{(1)}(i) = d^{(0)}(2i+1)$	$M_{\text{symb}}^{\text{layer}} = M_{\text{symb}}^{(0)}/2$
3	1	$x^{(0)}(i) = d^{(0)}(3i)$ $x^{(1)}(i) = d^{(0)}(3i+1)$ $x^{(2)}(i) = d^{(0)}(3i+2)$	$M_{\text{symb}}^{\text{layer}} = M_{\text{symb}}^{(0)}/3$
4	1	$x^{(0)}(i) = d^{(0)}(4i)$ $x^{(1)}(i) = d^{(0)}(4i+1)$ $x^{(2)}(i) = d^{(0)}(4i+2)$ $x^{(3)}(i) = d^{(0)}(4i+3)$	$M_{\text{symb}}^{\text{layer}} = M_{\text{symb}}^{(0)}/4$
5	2	$x^{(0)}(i) = d^{(0)}(2i)$ $x^{(1)}(i) = d^{(0)}(2i+1)$ $x^{(2)}(i) = d^{(1)}(3i)$ $x^{(3)}(i) = d^{(1)}(3i+1)$ $x^{(4)}(i) = d^{(1)}(3i+2)$	$M_{\text{symb}}^{\text{layer}} = M_{\text{symb}}^{(0)}/2 = M_{\text{symb}}^{(1)}/3$
6	2	$x^{(0)}(i) = d^{(0)}(3i)$ $x^{(1)}(i) = d^{(0)}(3i+1)$ $x^{(2)}(i) = d^{(0)}(3i+2)$ $x^{(3)}(i) = d^{(1)}(3i)$ $x^{(4)}(i) = d^{(1)}(3i+1)$ $x^{(5)}(i) = d^{(1)}(3i+2)$	$M_{\text{symb}}^{\text{layer}} = M_{\text{symb}}^{(0)}/3 = M_{\text{symb}}^{(1)}/3$
7	2	$x^{(0)}(i) = d^{(0)}(3i)$ $x^{(1)}(i) = d^{(0)}(3i+1)$ $x^{(2)}(i) = d^{(0)}(3i+2)$ $x^{(3)}(i) = d^{(1)}(4i)$ $x^{(4)}(i) = d^{(1)}(4i+1)$ $x^{(5)}(i) = d^{(1)}(4i+2)$ $x^{(6)}(i) = d^{(1)}(4i+3)$	$M_{\text{symb}}^{\text{layer}} = M_{\text{symb}}^{(0)}/3 = M_{\text{symb}}^{(1)}/4$
8	2	$x^{(0)}(i) = d^{(0)}(4i)$ $x^{(1)}(i) = d^{(0)}(4i+1)$ $x^{(2)}(i) = d^{(0)}(4i+2)$ $x^{(3)}(i) = d^{(0)}(4i+3)$ $x^{(4)}(i) = d^{(1)}(4i)$ $x^{(5)}(i) = d^{(1)}(4i+1)$ $x^{(6)}(i) = d^{(1)}(4i+2)$ $x^{(7)}(i) = d^{(1)}(4i+3)$	$M_{\text{symb}}^{\text{layer}} = M_{\text{symb}}^{(0)}/4 = M_{\text{symb}}^{(1)}/4$

图 2: Codeword-to-layer mapping for spatial multiplexing

3.2 程序结构修改情况

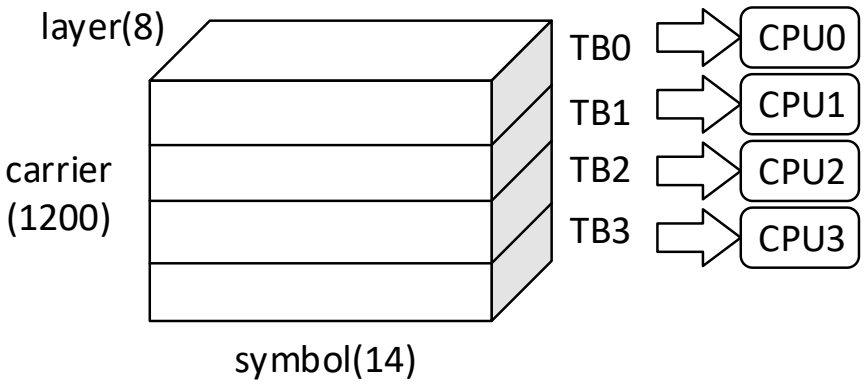

图 3: 原结构

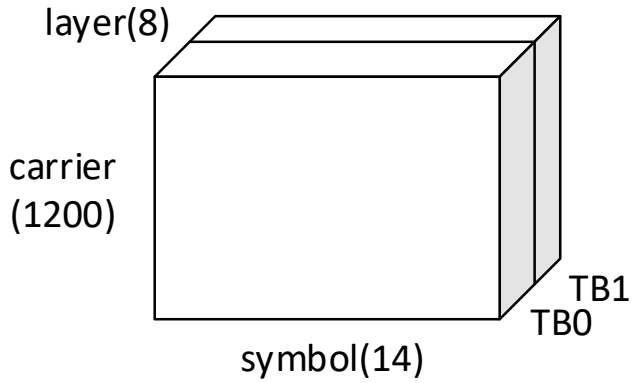

图 4: 协议要求结构

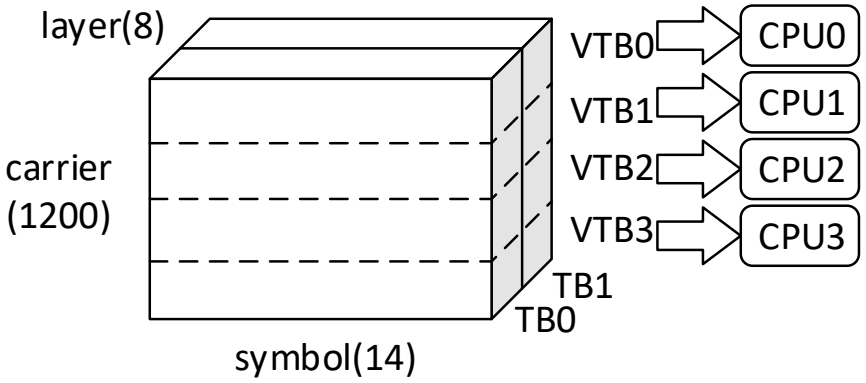

图 5: 修改后结果

4 根据新的资源映射结构修改程序

4.1 用多线程数据结构编写单线程程序

```
for (indx_tb = 0; indx_tb < tb_num; indx_tb++)
    for (indx_vtb = 0; indx_vtb < MAX_VTB_NUM; indx_vtb++)
        for (indx_cb = 0; indx_cb < encoder_t[indx_tb][indx_vtb].C; indx_cb++)
        {
            /* 1.2 code block segment */
            gettimeofday(&tbin, NULL);
            nr15_ldpc_simd_t *he = &encoder_t[indx_tb][indx_vtb];
            nr15_fec_ldpc_simd_decbs_scb(tx_data[indx_tb][indx_vtb], he, he->cbs_bits, indx_cb);
            gettimeofday(&tend, NULL);
            time_filter.tx_cbs += (tend.tv_sec - tbin.tv_sec) + (tend.tv_usec - tbin.tv_usec) / 1e6;

            /* 1.3 LDPC encoder */
            gettimeofday(&tbin, NULL);
            nr15_fec_ldpc_simd_encoder_scb(he->cbs_bits + indx_cb * he->K, he,
                                           he->coded_bits + indx_cb * (he->N + 2 * he->Z_c));
            gettimeofday(&tend, NULL);
            time_filter.tx_encode += (tend.tv_sec - tbin.tv_sec) + (tend.tv_usec - tbin.tv_usec) / 1e6;

            /* 1.4 rate matching */
            gettimeofday(&tbin, NULL);
            nr15_fec_ldpc_simd_rate_matching_scb(he->coded_bits, he, rmed_bits[indx_tb][indx_vtb], indx_cb);
            gettimeofday(&tend, NULL);
            time_filter.tx_rm += (tend.tv_sec - tbin.tv_sec) + (tend.tv_usec - tbin.tv_usec) / 1e6;
        }
```

图 6: 发送端的单线程模块

```
for (indx_vtb = 0; indx_vtb < MAX_VTB_NUM; indx_vtb++)
{
    if (indx_vtb == 0)
        start_sb = (CARR_NUM - vld_carr_num) / SB_CARR_NUM;
    for (indx_sb = start_sb; indx_sb < VTB_SB_NUM; indx_sb++)
    {
        /* 2.0 Get Source Block */
        nr5g_get_sb_sig(rx_sb_sig[indx_vtb][indx_sb], packed_rx_sig, SYM_NUM, CARR_NUM, RX_ANT_NUM,
                        indx_vtb * VTB_SB_NUM + indx_sb, MAX_VTB_NUM * VTB_SB_NUM);

        /* 2.1 Channel Estimate */
        gettimeofday(&tbin, NULL);
        est_sigma2[indx_vtb][indx_sb] = ChannelEstimator_LS(rx_sb_sig[indx_vtb][indx_sb],
                                                            pilot_sb_sym[total_layer_num - 1][indx_vtb * VTB_SB_NUM + indx_s
                                                            pilot_symb_indx, total_layer_num, RX_ANT_NUM, SB_CARR_NUM,
                                                            PILOT_SYM_NUM, SYM_NUM, inter_freq[total_layer_num - 1],
                                                            est_ch[indx_vtb][indx_sb]);
        gettimeofday(&tend, NULL);
        time_filter.rx_chest += (tend.tv_sec - tbin.tv_sec) + (tend.tv_usec - tbin.tv_usec) / 1e6;

        /* 2.2 Signal Detection */
        gettimeofday(&tbin, NULL);
        CalSymb_mmse(est_ch[indx_vtb][indx_sb], rx_sb_sig[indx_vtb][indx_sb], RX_ANT_NUM, SB_CARR_NUM,
                     total_layer_num, SYM_NUM, sqrt(est_sigma2[indx_vtb][indx_sb]), 1, 1,
                     detected_sig[indx_vtb][indx_sb], symb_var[indx_vtb][indx_sb],
                     detected_sinr[indx_vtb][indx_sb]);
        gettimeofday(&tend, NULL);
        time_filter.rx_sigdtct += (tend.tv_sec - tbin.tv_sec) + (tend.tv_usec - tbin.tv_usec) / 1e6;
    }
}
```

图 7: 接收端的单线程模块