组会报告

徐益

2018 年 8 月 2 日

1 工作内容

1. 学习 VTune 基本操作
2. 实现 LDPC layered 译码算法。
3. 学习

2 LDPC layered 代码结构

2.1 原 layered 译码数据结构

```
typedef struct check_node
{
        int8_t degree;          // number of connective variable nodes
        int16_t *index;         // index of connective variable nodes
        float* message;         // message from cn to vn
        float* last_message;    // last message from cn to vn
        float* vn_message;      // message from vn to cn
} check_node;
```

2.2 现 layered 译码数据结构

```
typedef struct check_node
{
        int8_t degree;          // number of connective variable nodes
        int16_t *index;         // index of connective variable nodes
        float* message;         // message from cn to vn
} check_node;
```

2.3 节点更新模块的封装

```
void update_message_layered_oms(float* cn_message,
                                int16_t* index,
                                int8_t degree,
                                float beta,
                                float* llr)
```

3 性能测试测试

表 1: 不同译码算法的吞吐量 ($SNR = 3.0dB, N = 25344, R = 0.5, I_{max} = 100$)

scheduling	BP	MS	NMS	OMS
flooding	120.65kbps	296.15kbps	265.81kbps	299.57kbps
layered	312.95kbps	751.47kbps	744.53kbps	855.86kbps

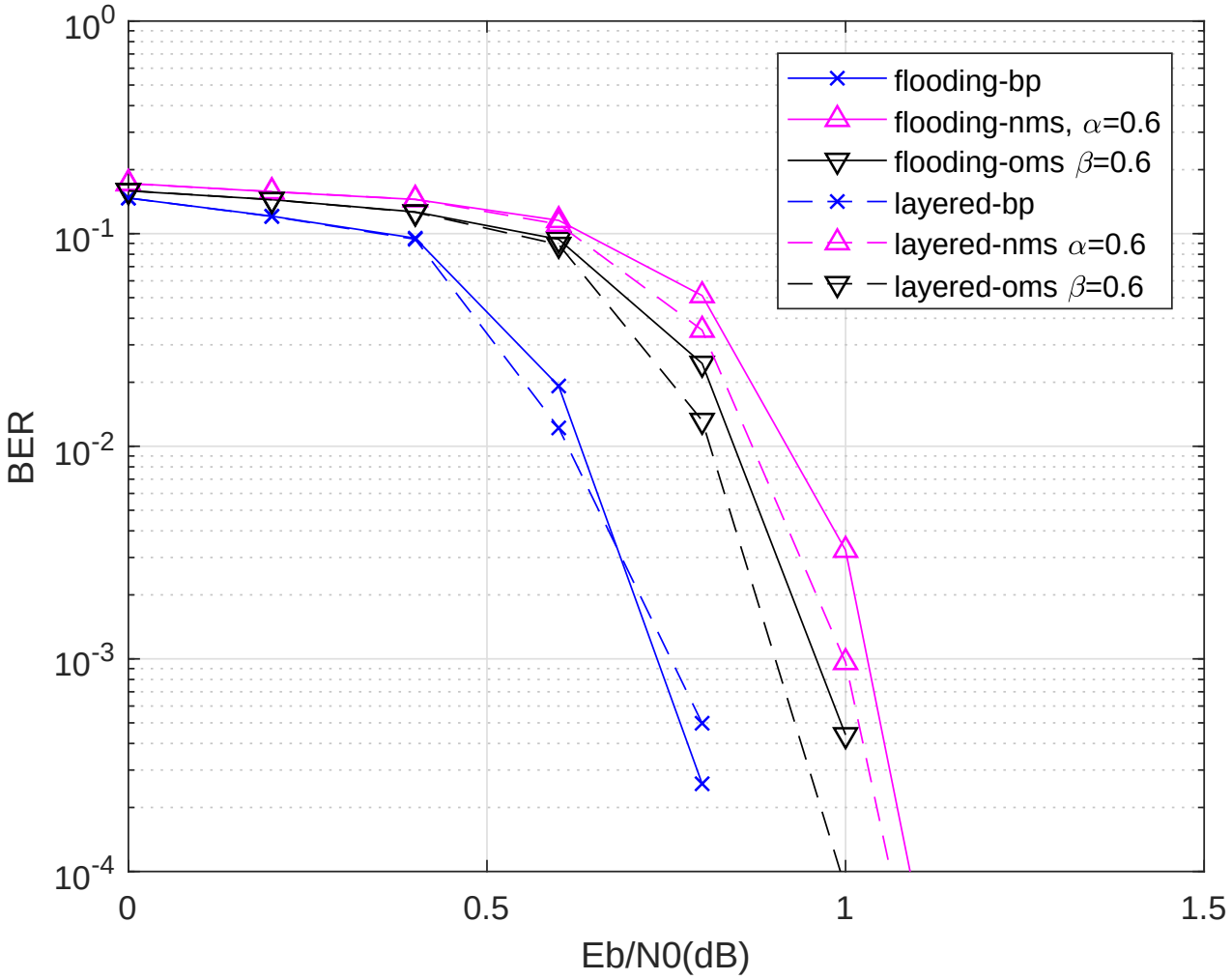

图 1: 不同译码算法的误码性能对比 ($B = 8448, R = 0.5, I_{max} = 50$)

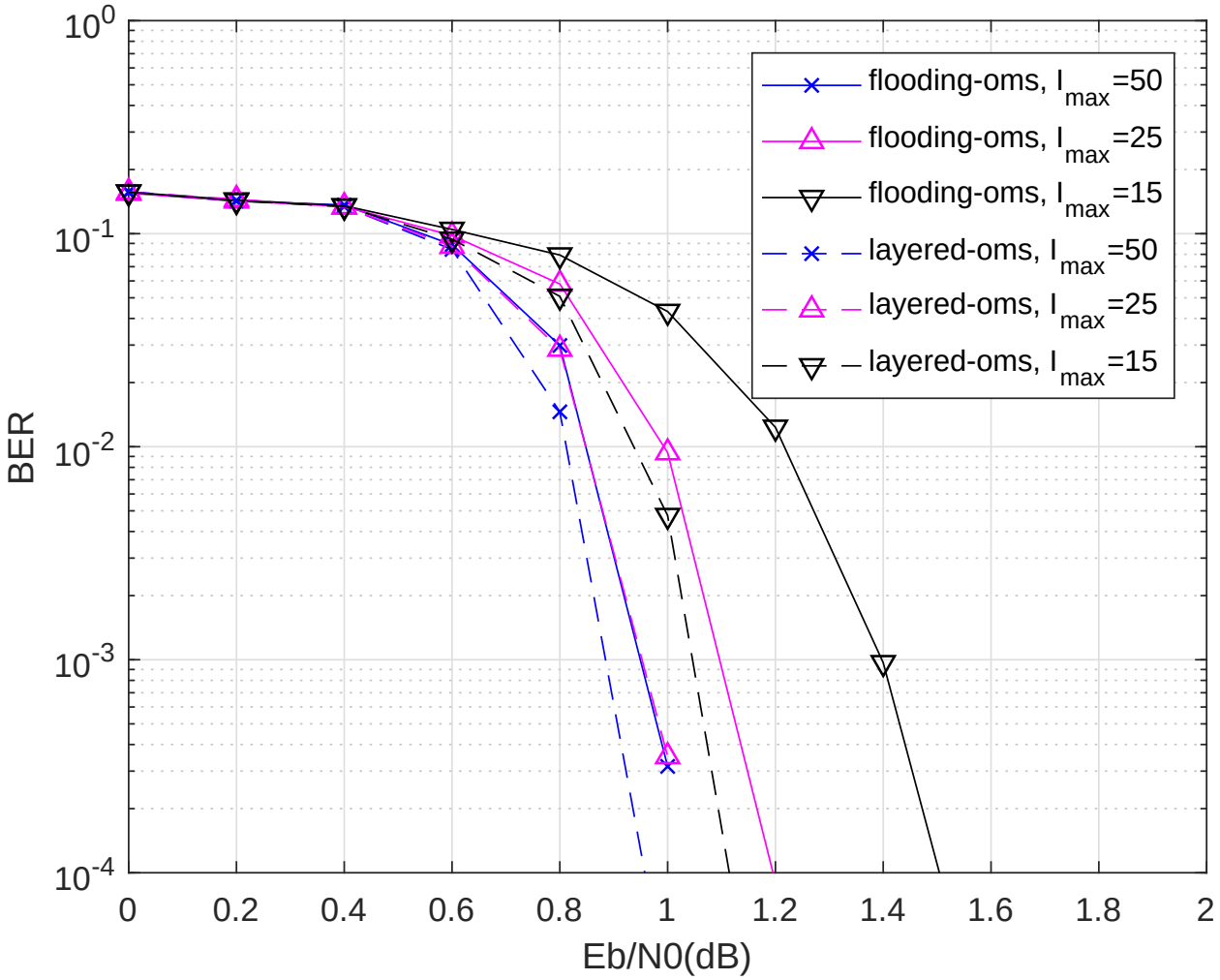

图 2: 不同译码算法的误码性能对比 ($B = 8448, R = 0.5, \beta = 0.6$)

4 下阶段计划

1. 将 message 数据类型改为 int_8
2. 尝试 SIMD 改写