组会报告

徐益

2018 年 8 月 20 日

1 工作内容

1. 学习 High-Throughput Multi-Core LDPC Decoders Based on x86 Processor。
2. 学习相关代码。

2 论文对 layered-ms 算法的改进

2.1 原数据结构

```
int t_deg_cn[n-k] = { 4, 3, 4, 3, 4 };

unsigned short indicies[ ] = {
        0, 1, 2, 3, // VN nodes for C0
        3, 4, 5,    // VN nodes for C1
        1, 4, 6, 7, // VN nodes for C2
        0, 3, 6,    // VN nodes for C3
        0, 2, 3, 6  // VN nodes for C4
};

// decoder description (init + iteration loop)
for(n=0; n<C; n++)
  unsigned char deg_cn = t_deg_cn[n];
  for(i = 0; i < deg_cn; i++){ /* COMPUTE CN value */ }
  for(i = 0; i < deg_cn; i++){ /* UPDATE MSG&VN values */ }
}
// end of decoding process description (hard decision)
```

图 1: Naive decoder kernel description using constant arrays

空间占用情况：

$$\Delta = 4 \times n + 4 \times m + 2 \times m + (n - k). \tag{1}$$

2.2 msg 类型优化

从 float 变成 int8_t
空间占用情况：

$$\Delta = n + m + 2 \times m + (n - k). \tag{2}$$

2.3 基于交织的并行计算

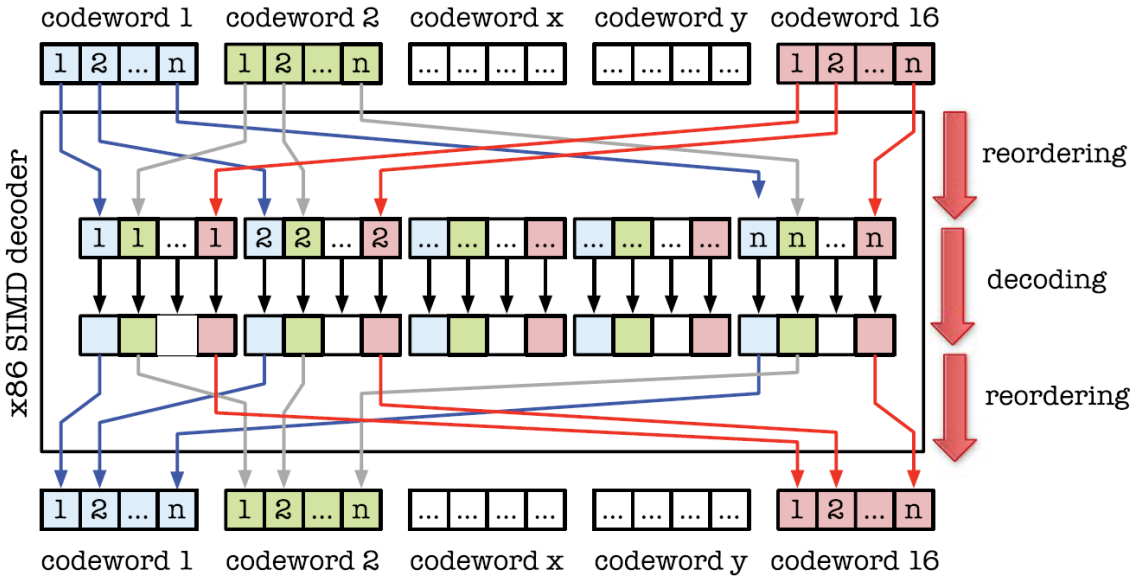

图 2: Data interleaving and desinterleaving processes

空间占用情况：

$$\Delta = q \times n + q \times m + 2 \times m + (n - k). \tag{3}$$

2.4 重新排布校验矩阵

$$\begin{array}{c} \\ C_0[\delta_0=4] \\ C_1[\delta_1=3] \\ C_2[\delta_2=4] \\ C_3[\delta_3=3] \\ C_4[\delta_4=4] \end{array} \begin{array}{c} \begin{array}{cccccccc} V_0 & V_1 & V_2 & V_3 & V_4 & V_5 & V_6 & V_7 \end{array} \\ \begin{pmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 \end{pmatrix} \end{array} \Rightarrow \begin{array}{c} \\ C_0[\delta_0=4] \\ C_2[\delta_1=4] \\ C_4[\delta_2=4] \\ C_1[\delta_3=3] \\ C_3[\delta_4=3] \end{array} \begin{array}{c} \begin{array}{cccccccc} V_0 & V_1 & V_2 & V_3 & V_4 & V_5 & V_6 & V_7 \end{array} \\ \begin{pmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \end{pmatrix} \end{array} \Rightarrow \begin{array}{c} \\ C_0[\delta_0=4] \\ C_4[\delta_4=4] \\ C_2[\delta_2=4] \\ C_1[\delta_1=3] \\ C_3[\delta_3=3] \end{array} \begin{array}{c} \begin{array}{cccccccc} V_0 & V_1 & V_2 & V_3 & V_4 & V_5 & V_6 & V_7 \end{array} \\ \begin{pmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \end{pmatrix} \end{array}$$

图 3: CN Computation Reordering

空间占用情况：

$$\Delta = q \times n + q \times m + 2 \times m. \tag{4}$$

2.5 其他

1. 预先计算好 En 地址;
2. 使用最新的指令集;
3. 多核计算。

3 代码学习

Algorithm 1. Horizontal TDMP Min-Sum algorithm

Kernel 1: Initialization
for all $m \in C, n \in \Psi(m)$ **do**
  $L_{mn}^{(0)} = 0$
end for
▷ *Process iter_max decoding iterations*
for all $t = 1 \rightarrow (iter\_max)$ **do**
  Kernel 2: For each check node in the code
  for all $m \in C$ **do**
    ▷ *Compute* L_{nm} *message*
    for all $n \in \Psi(m)$ **do**
      $L_{nm}^{(t)} = E_n - L_{mn}^{(t-1)}$
    end for
    ▷ *Compute* L_{mn} *message*
    for all $n \in \Psi(m)$ **do**
      $sign(L_{mn}^{t}) = \left[\prod_{(n' \in \Psi(m)/n)} sign(L_{n'm}^{(t)})\right]$
      $|L_{mn}^{t}| = \left[\min_{(n' \in \Psi(m)/n)} |L_{n'm}^{(t)}|\right]$
    end for
    ▷ *Immediately update* E_n
    for all $n \in \Psi(m)$ **do**
      $E_n = L_{nm}^{t} + L_{mn}^{t}$
    end for
  end for
end for
Kernel 3: Hard decision
for all $n \in V$ **do**
  $\hat{c}_n = \begin{cases} 0 & \text{if } E_n \leq 0 \\ 1 & \text{if } E_n > 0 \end{cases}$
end for

4 存在问题

5 下阶段计划

1. 使程序正常运行；
2. 尝试与原仿真程序结合。