마이크로프로세서응용

<16조, final>

조원 : 201810528 고려욱

201810845 박종혁

1. LDPC

1) 동작사진

```
<LDPC decoder>

Measured Accuracy : NSR(dB) = -inf

-----Benchmarking Start-----
Case 0: LDPC Reference
Nr,          Max,          Min,      Average,    Fltr Avg,  Fltr_Avg(ms)
 5,     41537052,     41526981,     41532189,     41532305,      124.597
Case 1: LDPC Optimization
Nr,          Max,          Min,      Average,    Fltr Avg,  Fltr_Avg(ms)
 5,      7628820,      7620078,      7623049,      7622116,       22.866
----Benchmarking Complete----

Optimized LDPC Decoder is x5.44 faster than Reference
```

2) 전체 코드

```
void ldpcd_opt(float(*NLLR)[N], float** opt_out) {

    ////////////////////////
    // Edit code below!! //

    //동적할당은 정확성만 낮아지
    float * Zxn = (float *)calloc((Z*Mp)*(Z*Np),sizeof(float));
    float * Zn = (float *)calloc((Z * Np),sizeof(float));
    float * Lxn = (float *)calloc((Z*Mp)*(Z*Np),sizeof(float));

    float Zn[Z * Np] = { 0, };  // final Value L
    int tp[Z * Mp][Z * Np] = { 0, };
    float a = 0;
    int count[Z * Np] = { 0, };
    int ct;

    for (int p = 0; p < WC; p++) {
        for (int j = 0; j < Z * Np; j++) {
            Zn[j] = NLLR[p][j];
        }
        float Zxn[Z * Mp][Z * Np] = { 0, };      //초기화 시마다 배열 새로 선언
        float Lxn[Z * Mp][Z * Np] = { 0, };      //초기화 시마다 배열 새로 선언

        for (int k = 0; k < iter; k++) {
            for (int j = 0; j < M; j++) {
                float min1 = 500;  //가장 최소값
                float min2 = 500;  //두번째 최소값
                int minx = 0;        //첫번째 최소값 위치 저장
                int sgn = 1;         //부호저장
                int t = 0;           //H배열이 1인 배열의 i값 저장
                if (k == 0) {
                    ct = 0;
                    for (int i = 0; i < N; i++) {
                        if (H[j][i] == 1)
                            tp[j][ct++] = i;
                    }
                    count[j] = ct;
                }
                for (int r = 0; r < count[j]; r++) {
                    t = tp[j][r];
                    Zxn[j][t] = Zn[t] - Lxn[j][t];
                    if (Zxn[j][t] < 0) {
                        sgn = sgn * -1;
                    }
                    a = fabs(Zxn[j][t]);
                    if (min1 > a) {
                        min2 = min1;
```

```
                    min1 = a;
                    minx = t;
                }
                else if (min2 > a)
                    min2 = a;
            }

            //sel step
            for (int r = 0; r < count[j]; r++) {
                t = tp[j][r];
                if (minx == t) {
                    if (min2 > Offset) {
                        Lxn[j][t] = min2 - Offset;
                    }
                    else {
                        Lxn[j][t] = 0;
                    }
                }
                else {
                    if (min1 > Offset) {
                        Lxn[j][t] = min1 - Offset;
                    }
                    else {
                        Lxn[j][t] = 0;
                    }
                }
                if (Zxn[j][t] >= 0) {
                    Lxn[j][t] = Lxn[j][t] * sgn;
                }
                else
                    Lxn[j][t] = Lxn[j][t] * sgn * -1;

                Zn[t] = Zxn[j][t] + Lxn[j][t];
            }
        }
    }

    /* output before Hard Decision */
    for (int i = 0; i < MAX_LENGTH; i++)
        opt_out[p][i] = Zn[i];
}

// Edit code above!! //
//////////////////////
}
```

3) 적용 아이디어 설명

3-1) 적용 아이디어

(1) Loop Optimization 적용

```
for (int r = 0; r < count[j]; r++) {
    t = tp[j][r];
    if (minx == t) {
        if (min2 > Offset) {
            Lxn[j][t] = min2 - Offset;
        }
        else {
            Lxn[j][t] = 0;
        }
    }
    else {
        if (min1 > Offset) {
            Lxn[j][t] = min1 - Offset;
        }
        else {
            Lxn[j][t] = 0;
        }
    }
    if (Zxn[j][t] >= 0) {
        Lxn[j][t] = Lxn[j][t] * sgn;
    }
    else
        Lxn[j][t] = Lxn[j][t] * sgn * -1;

    Zn[t] = Zxn[j][t] + Lxn[j][t];
}
```

- Ref 코드의 For loop를 합치고 순서를 바꿔서 최소한의 for loop를 동작하게 만듬. For-loop를 줄이면서 branch instruction을 줄일 수 있음.
  <prilimary 기준으로 약 69ms 성능개선>

(2) 초기화 및 변수 선언을 for loop안쪽에서 동작

```
for (int p = 0; p < WC; p++) {
    for (int j = 0; j < Z * Np; j++) {
        Zn[j] = NLLR[p][j];
    }
    float Zxn[Z * Mp][Z * Np] = { 0, };     //초기화 시마다 배열 새로 선언
    float Lxn[Z * Mp][Z * Np] = { 0, };     //초기화 시마다 배열 새로 선언

    for (int k = 0; k < iter; k++) {
        for (int j = 0; j < M; j++) {
            float min1 = 500;  //가장 최소값
            float min2 = 500;  //두번째 최소값
            int minx = 0;       //첫번째 최소값 위치 저장
            int sgn = 1;        //부호저장
            int t = 0;          //H배열이 1인 배열의 i값 저장
```

미리 밖에서 선언한 변수들을 가져오는 것이 아니라 새로 생성하여 저장
이를 통해 초기화에 대한 코드를 없애고 assembly상에서 변수 위치에 접근하는데 들어가는 시간을 줄일 수 있음.
<약 3ms 성능개선>

(3) 외부 글로벌 H배열에 대한 접근을 행 loop안에서 한번 만 동작하여 조건 지정한 것 최소한으로 동작할 수 있도록 함

```
if (k == 0) {
    ct = 0;
    for (int i = 0; i < N; i++) {
        if (H[j][i] == 1)
            tp[j][ct++] = i;
    }
    count[j] = ct;
```

Tp2차원배열에 H배열에서 지정한 행 안에 값이 1인 열의 값들을 담아낸다. K가 0일때만 동작하여 계속 반복시에 똑 같은 행의 값에 대한 접근을 줄여주는 역할을 한다. 반복문을 돌며 Ct변수에 총 행의 수를 담고, count배열에 그 열의 총 개수를 담아 뒤에서 참조 할 때 편하게 적용한다.

이를 통해 기존의 ref코드에서 계속해서 해당 cell이 H matrix에서 1인지 검사하는데 들어가는 시간을 최소한으로 줄일 수 있다.

```
for (int r = 0; r < count[j]; r++) {
    t = tp[j][r];
```

접근은 해당열에서 H배열의 값이 1을 가지는 행들만 순서대로 가져와 지정한 LDPC decoding을 실행한다.

<약 35ms 성능개선>

(4) 임시변수 활용

```
for (int r = 0; r < count[j]; r++) {
    t = tp[j][r];
    a = Lxn[j][t];
    if (minx == t) {
        if (min2 > Offset) {
            a = min2 - Offset;
        }
        else {
            a = 0;
        }
    }
    else {
        if (min1 > Offset) {
            a = min1 - Offset;
        }
        else {
            a = 0;
        }
    }
    if (Zxn[j][t] >= 0) {
        a = a * sgn;
    }
    else
        a = a * sgn * -1;
    Lxn[j][t] = a;
    Zn[t] = Zxn[j][t] + Lxn[j][t];
}
```

- 임시변수에 계산 시 활용할 배열의 값을 저장하여 계산마다 배열에 접근하지 않도록 함.

<약 1ms성능개선>