마이크로프로세서응용

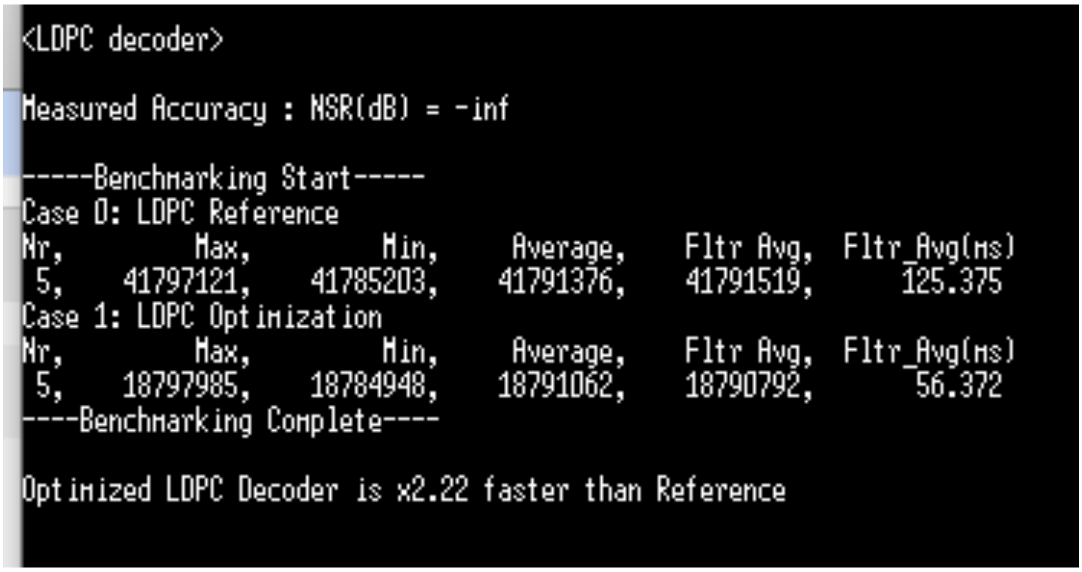
<16조, preliminary>

조원 : 201810528 고려욱

201810845 박종혁

1. LDPC

1) 동작사진



2) 적용 아이디어 설명

Loop Optimization 적용

- For loop 합쳐서 For loop 최소한으로 동작하도록 유도
- For loop 순서 바꿔서 여러 개 For loop 한번에 동작하도록 유도

Cache Optimize 적용

- 동일한 matrix 연속해서 접근 가능하도록 코드의 순서 변경

3) 보완방향

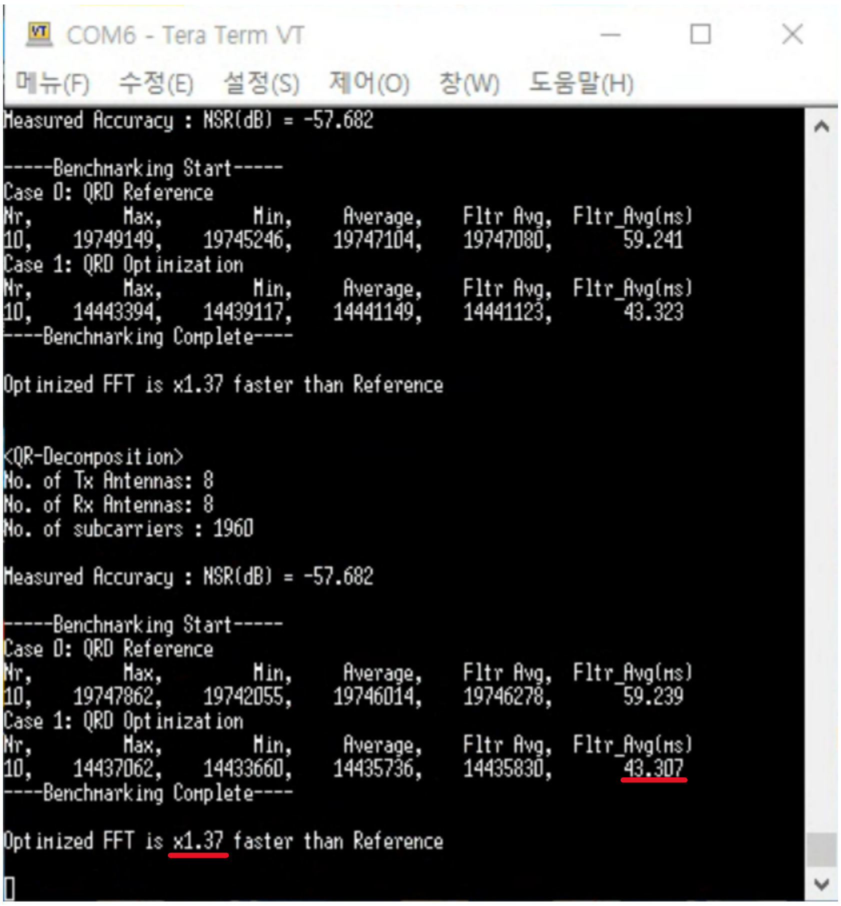
Tiling 적용

H matrix 접근을 한번에 통일하여 동작하도록 유도

최소합 알고리즘, LLR-SPA 알고리즘 적용

2. QRD

1) 동작사진



2) 적용 아이디어 설명

> Loop Unrolling

```
for(i = 0; i < NSC * NRX * NTX; i+=2)          // loop unrolling
{
    Q[i] = H[i];
    Q[i+1] = H[i+1];
}
```

> Dave Eberly’s fast inverse square root algorithm

```c
// invert sqrt
// Dave Eberly's FastinverseSqrt Algorithm
tmp_i = sq;
fHalf = 0.5f * sq;
t = *(int*)&sq;
t = 0x5f3759df - (t >> 1);
sq = *(float*)&t;
sq = sq*(1.5f - fHalf*sq*sq);

tmp_r = sq * tmp_i;      // tmp_r = fastSqrt(sq);
```

- 기존의 <math.h> 의 sqrt 함수보다 더 빠른 fast inverse square root 알고리즘을 적용시켰다.

> Frequency reduction(Temporary Variables)

```c
// Compute Row
for(k = j + 1; k < NTX; k++)
{
    tmp_r = 0, tmp_i = 0;        // Frequency Reduction
    for(l = 0; l < NRX; l++)
    {
        tmp_r += (Q[i * NTX * NRX + l * NTX + j].real * Q[i * NTX * NRX + l * NTX + k].real
                                    + Q[i * NTX * NRX + l * NTX + j].img * Q[i * NTX * NRX + l * NTX + k].img);
        tmp_i += (Q[i * NTX * NRX + l * NTX + j].real * Q[i * NTX * NRX + l * NTX + k].img
                                    - Q[i * NTX * NRX + l * NTX + j].img * Q[i * NTX * NRX + l * NTX + k].real);
    }
    R[i * NTX * NTX + j * NTX + k].real = tmp_r;
    R[i * NTX * NTX + j * NTX + k].img = tmp_i;
}
```

```c
// Normalization Column - Temporary variable
R[i * NTX * NTX + j * NTX + j].real = tmp_r;
for(k = 0; k < NRX; k++)
{
    Q[i * NTX * NRX + k * NTX + j].real /= tmp_r;
    Q[i * NTX * NRX + k * NTX + j].img /= tmp_r;
}
```

- 루프 내에서 변수화 시킬 수 있는 것을 찾아, 변수화 시켜주었다.

> **Frequency Reduction(Temporary Variables), Loop unrolling, Loop order change**

```
// Update Column - Temporary variable & loop order change
for(l = 0; l < NRX; l++)
{
    tmp_r = Q[i * NTX * NRX + l * NTX + j].real;        // Frequency Reduction
    tmp_i = Q[i * NTX * NRX + l * NTX + j].img;

    for(k = (j + 1); k < 7; k+=2)          // loop unrolling
    {
        Q[i * NTX * NRX + l * NTX + k].real = Q[i * NTX * NRX + l * NTX + k].real
                                      - (R[i * NTX * NTX + j * NTX + k].real * tmp_r
                                      - R[i * NTX * NTX + j * NTX + k].img * tmp_i);
        Q[i * NTX * NRX + l * NTX + k].img = Q[i * NTX * NRX + l * NTX + k].img
                            - (R[i * NTX * NTX + j * NTX + k].real * tmp_i
                                + R[i * NTX * NTX + j * NTX + k].img * tmp_r);

        Q[i * NTX * NRX + l * NTX + (k+1)].real = Q[i * NTX * NRX + l * NTX + (k+1)].real
                            - (R[i * NTX * NTX + j * NTX + (k+1)].real * tmp_r
                                - R[i * NTX * NTX + j * NTX + (k+1)].img * tmp_i);
        Q[i * NTX * NRX + l * NTX + (k+1)].img = Q[i * NTX * NRX + l * NTX + (k+1)].img
                            - (R[i * NTX * NTX + j * NTX + (k+1)].real * tmp_i
                                + R[i * NTX * NTX + j * NTX + (k+1)].img * tmp_r);
    }

    if(j % 2 == 0){          // if((8-(j+1)) % 2 == 1)
        Q[i * NTX * NRX + l * NTX + 7].real = Q[i * NTX * NRX + l * NTX + 7].real
                                  - (R[i * NTX * NTX + j * NTX + 7].real * tmp_r
                                  - R[i * NTX * NTX + j * NTX + 7].img * tmp_i);
        Q[i * NTX * NRX + l * NTX + 7].img = Q[i * NTX * NRX + l * NTX + 7].img
                                  - (R[i * NTX * NTX + j * NTX + 7].real * tmp_i
                                  + R[i * NTX * NTX + j * NTX + 7].img * tmp_r);
    }
}
```

- 여러가지 loop optimization을 적용시켰다.

> **Loop Tiling**

```
#ifndef min
#define min(a,b) (((a) < (b)) ? (a) : (b))
#endif
```

```
for(ii = 0; ii < NSC; ii+=8)     // loop tiling
for(jj = 0; jj < NTX; jj+=8)
for(i = ii; i < min(NSC, ii+8); i++)
{
    for(j = jj; j < min(NTX, jj+8); j++)
    {
        sq = 0;
        // L2 Norm
```

- loop tiling을 위해, min(a,b)를 define 해주었다.

- tile size는 8로 설정하여 loop tiling을 진행했다.