

저전력컴퓨팅 2차 과제

컴퓨터학과 2020320026 권도혁

1. Thermal Modeling

특정 블록 Block(i, j)의 Thermal Circuit은 전기 회로와 유사하게 RC Circuit으로 표현될 수 있다. 이를 electricity와 thermality의 duality를 활용하여 방정식으로 표현할 수 있다. 먼저, RC circuit에서의 capacitor로 표현한 전류는 thermal equation에서 다음과 같이 표현될 수 있다

$$I = C \frac{dV}{dt} \rightarrow P = C_{th} \frac{\Delta T}{\Delta t}$$

그리고, RC circuit에서의 resistor로 표현한 전류는 thermal equation에서 다음과 같이 표현될 수 있다.

$$I = \frac{V_1 - V_2}{R} \rightarrow P = \frac{T_1 - T_2}{T_{th}}$$

이러한 관계를 이용하여 $P_{i,j}$ 와 $\Delta T_{i,j}$ 를 구할 수 있다. 먼저, $P_{i,j}$ 의 경우 주위 블록들과의 온도 차이와 저항으로 결정된다.

$$P_{i,j} = \frac{T_{i,j} - T_{i-1,j}}{R_1} + \frac{T_{i,j} - T_{i,j+1}}{R_2} + \frac{T_{i,j} - T_{i+1,j}}{R_3} + \frac{T_{i,j} - T_{i,j-1}}{R_4}$$

다음으로 $\Delta T_{i,j}$ 의 경우 특정 시간동안의 위에서 구한 전력 $P_{i,j}$ 와 커패시터 C 에 의해 결정된다.

$$\Delta T_{i,j} = \frac{P_{i,j} \cdot \Delta t}{C_{th}}$$

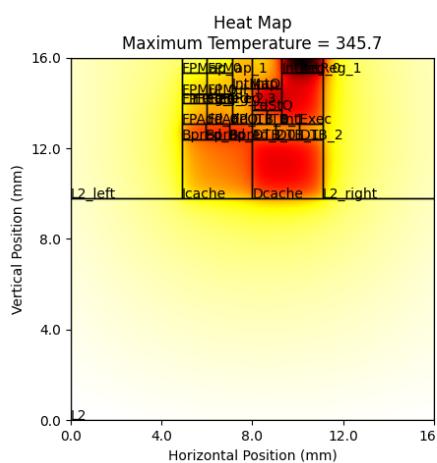
2. Model Granularity

Grid와 Block model에 대해 HotSpot을 실행했을 때, Steady-state와 Transient temperature는 다음과 같다.

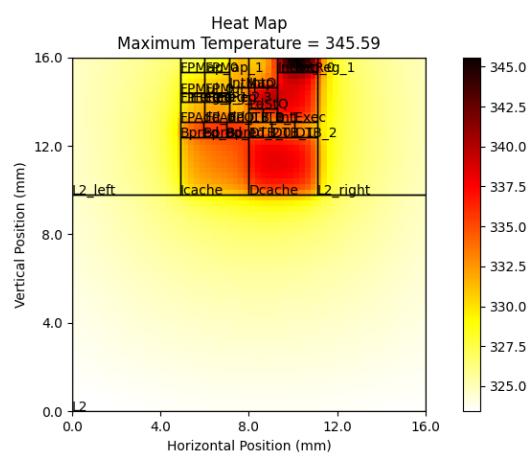
2-(1) Grid Model

a. Steady-state

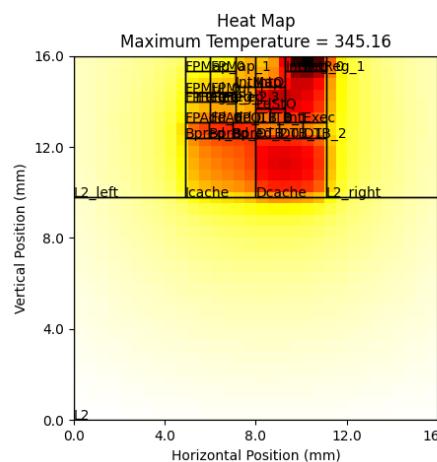
Grid model을 사용하기 위해 example2 디렉토리의 `run.sh` 파일을 사용했고, granularity를 바꾸기 위해 명령어 `../.../hotspot -c example.config -f ev6.flp -p gcc.ptrace -materials_file example.materials -model_type grid -grid_rows 128 -grid_cols 128 -steady_file outputs/gcc.steady -grid_steady_file outputs/gcc.grid.steady` 의 `-grid_rows` 와 `-grid_cols`의 값을 16X16에서 128X128까지 늘려가며 실행했다. 또한 die의 위치인 layer 0만을 사용하여 heatmap을 나타내었다.



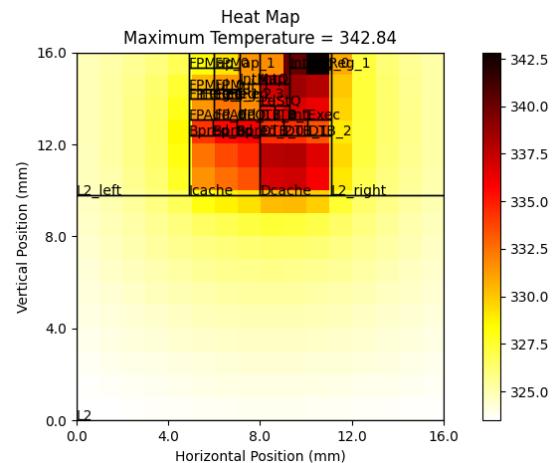
Heatmap for 128X128 Grid Model



Heatmap for 64X64 Grid Model



Heatmap for 32X32 Grid Model



Heatmap for 16X16 Grid Model

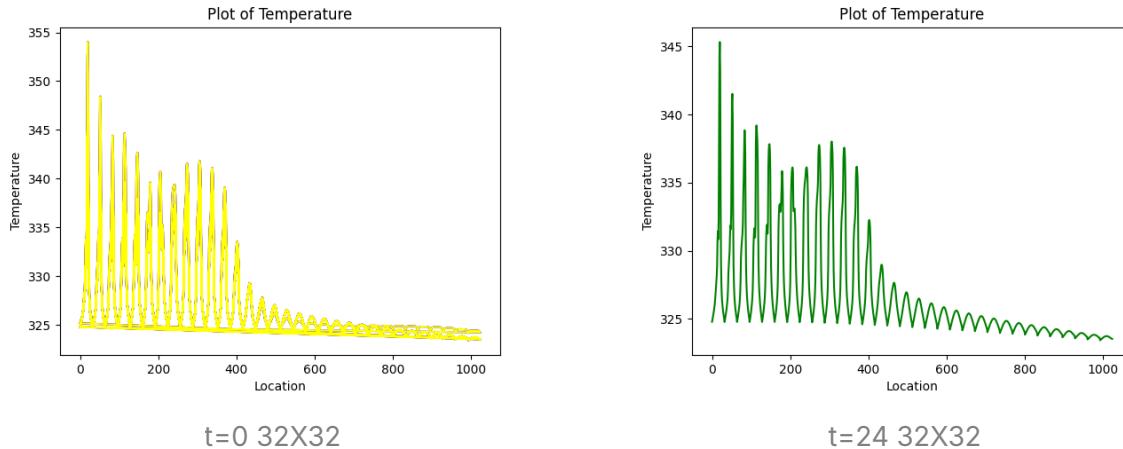
Granularity를 바꾸면서 실행했을 때, grid의 행과 열 개수가 많아질수록 더 정확한 히트맵을 표현할 수 있다. 이는 행과 열이 형성하는 단위의 크기가 작아질수록 온도를 측정하는 위치가 많아지기 때문에 더 정확한 온도 측정과 정밀한 시각화가 가능하다. 하지만 온도 측정의 위치가 많아진다는 것은 온도 측정의 횟수가 그만큼 많아지는 것을 의미하기 때문에 accuracy와 측정 횟수 간의 trade-off가 존재한다.

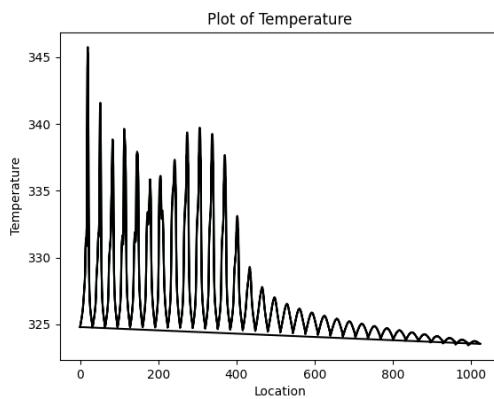
$$\text{Accuracy} \iff \# \text{ of Measurement(Computation)}$$

실제로, `run.sh` 파일을 실행할 때 행과 열의 개수가 많을수록 실행 시간이 더 오래 소모되었다.

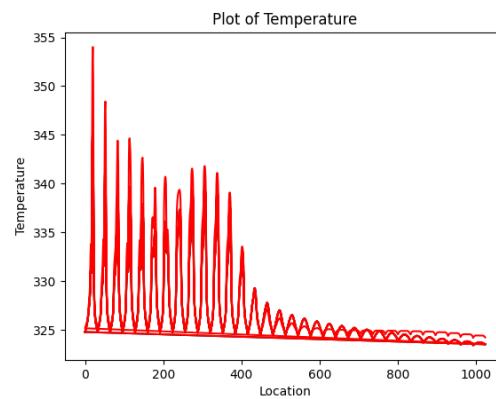
b. Transient

Transient의 경우 `split_grid_transient.py` 파일을 실행하여 각 layer 별로 `ttrace` 를 얻어서 layer 0 값만 사용하여 `t = 0`, `t = 24`, `t = 49`, `t = 74`, `t = 99` 에 따라 각각 plot을 만들어서 시간의 흐름에 따른 각 location의 측정 온도의 변화를 시각화하였다.

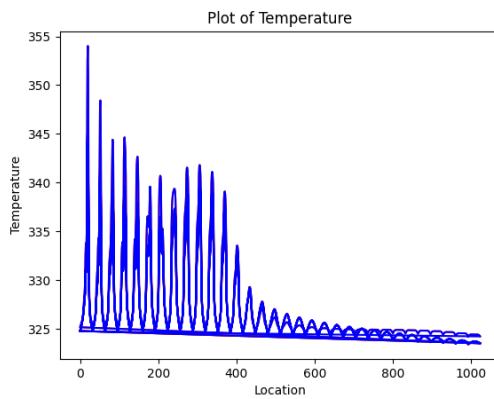




$t=49$ 32X32

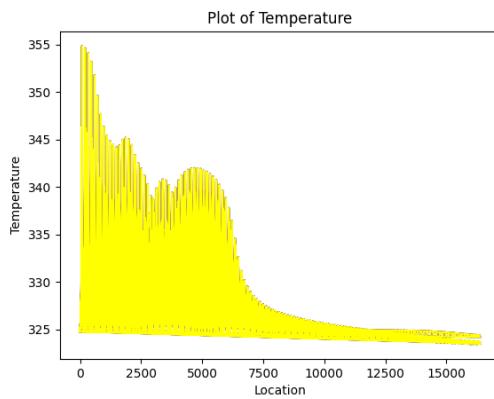


$t=74$ 32X32

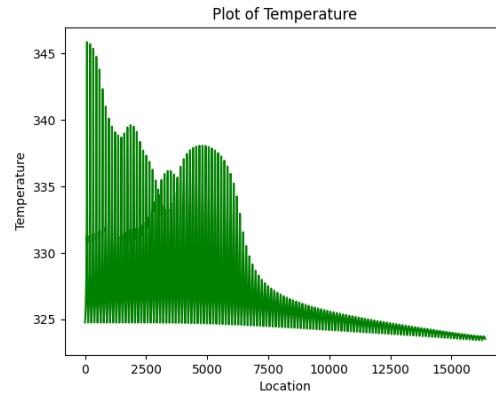


$t=99$ 32X32

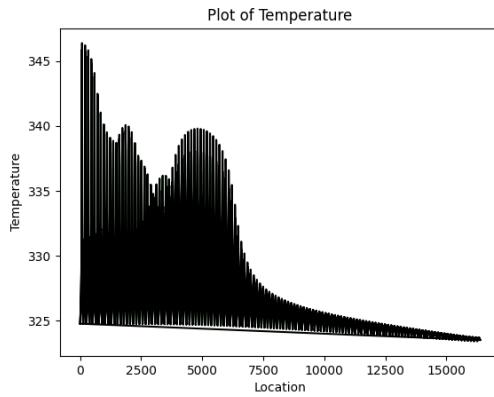
첫 번째 plot부터 다섯 번째 plot까지 grid model 32X32에 대한 각각 $t=0$ 부터 $t=99$ 까지의 plot이다.



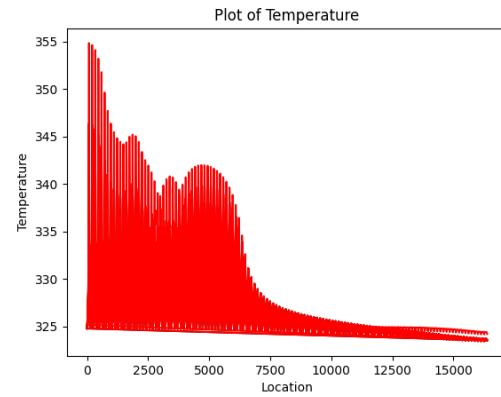
$t=0$ 128X128



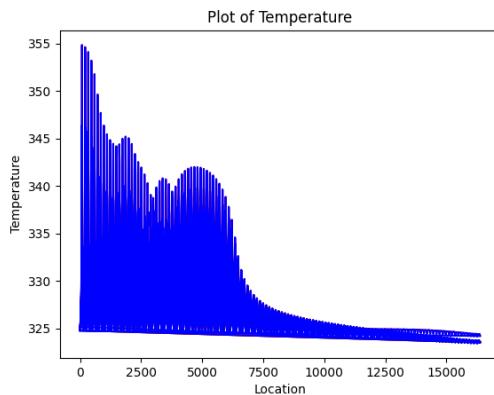
$t=24$ 128X128



t=49 128X128



t=74 128X128



t=99 128X128

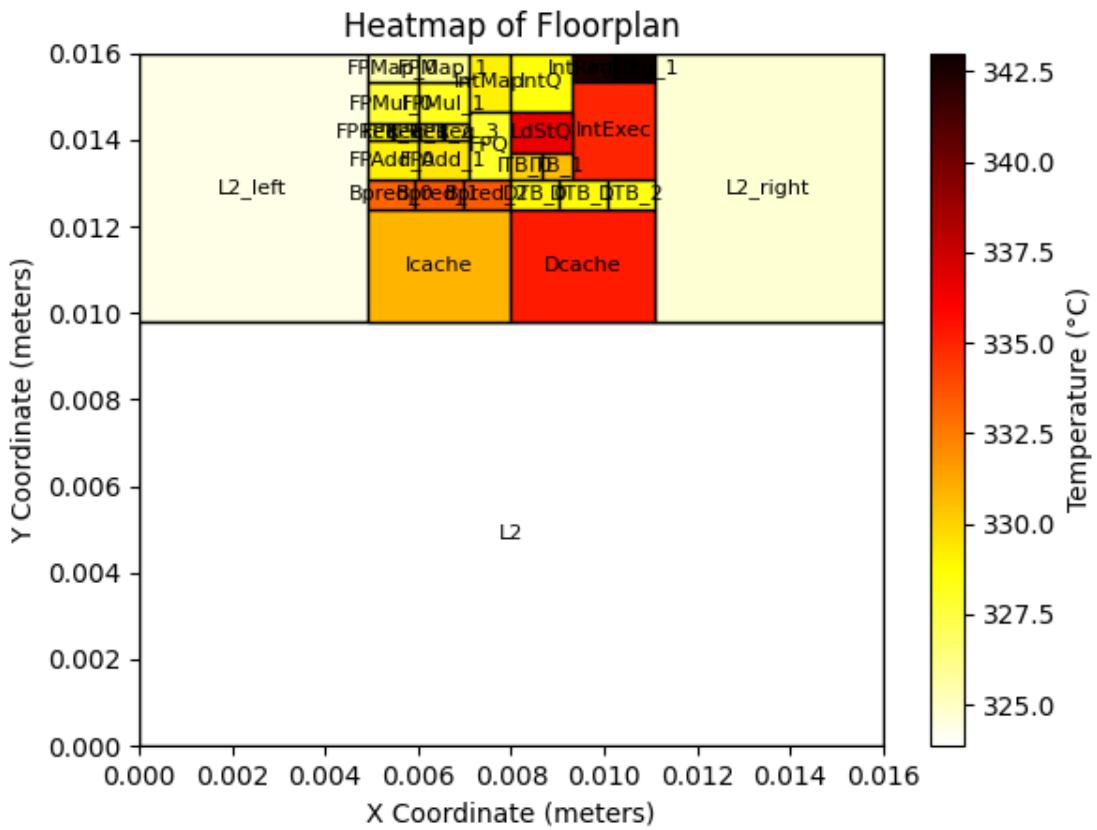
다음의 첫 번째부터 다섯 번째까지의 plot은 grid model 128X128에 대한 각각 t=0부터 t=99까지의 plot이다.

각각의 plot을 봤을 때 32X32 모델과 128X128 모델이 그리는 plot의 실루엣은 비슷하지만 128X128 모델이 훨씬 정교한 시각화를 보여주는 것을 확인할 수 있다.

2-(2) Block Model

Steady-state

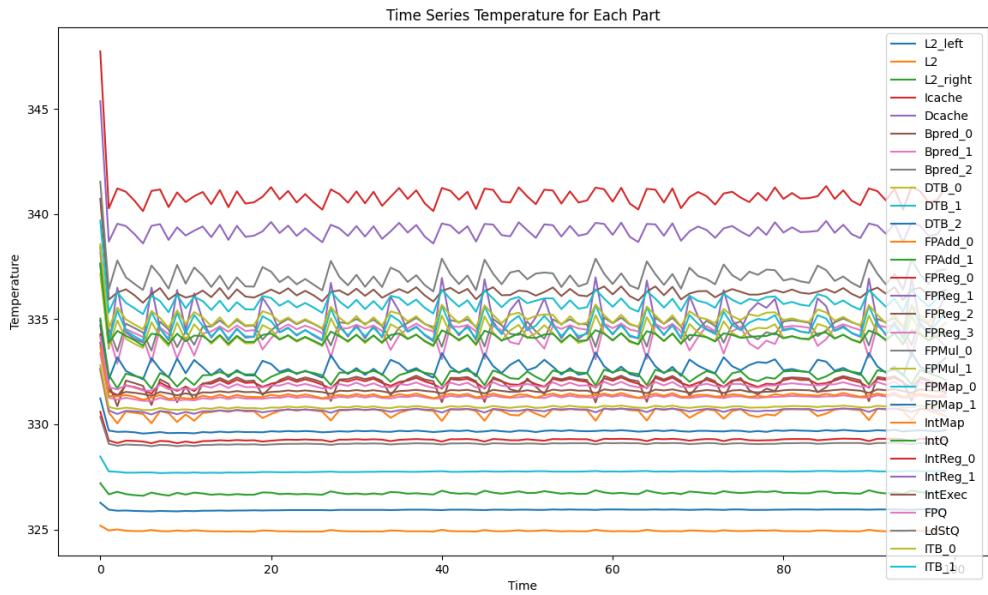
Block Model의 예시인 example1의 `run.sh` 를 사용하여 먼저 각 part의 steady 온도 정보를 얻었다. 그리고 `ev6.flp` 파일의 floorplan을 참고하여 각 part의 온도를 시각화된 floorplan에 heatmap으로 표시하였다. Grid Model과 마찬가지로 layer 0의 값을 사용하였다.



이를 나타내기 위해 `block.py`라는 새로운 코드를 작성하였고, 이는 `/scripts/grid_thermal_map.py` 파일을 참고하였다. Block Model의 총 계산 시간은 매우 빨랐지만, 각 part의 온도를 하나의 숫자로 표현하기 때문에 정확도는 낮다. 따라서 Block Model에서도 정확도와 속도 간의 trade-off가 있음을 확인할 수 있었다.

Transient

Grid Model을 transient 상황일 때 사용했던 `/example2` 를 사용하여 Block Model에서의 각 part 별로 시간에 따른 온도 변화를 시각화 해보았다. 이를 위해 `block_transient.py`라는 새로운 코드를 작성했고, 각 part 별 시간에 따른 온도는 다음과 같다.



Grid Model과 Block Model의 비교

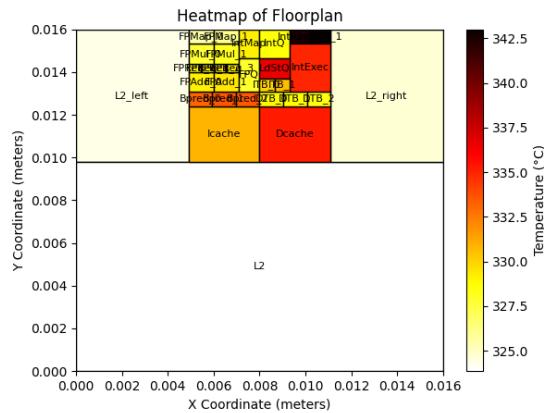
Grid Model의 경우 대체로 Block Model보다 계산 속도가 더 빨랐다. 이는 $N \times N$ Grid에서 총 N^2 번의 계산을 각각의 cell마다 해줘야 하지만, Block Model은 해당 floorplan이 가지고 있는 컴포넌트의 온도를 계산해주면 되기 때문이다. 그러나, Grid Model은 계산을 많이하는 대신 해당 위치에 대한 온도 정보는 Block Model보다 훨씬 정확하다는 장점이 있다. Grid Model의 경우에도 Grid의 granularity가 작은 경우 Block Model보다 계산 속도가 빠른 경우가 있지만, 이 경우에는 Block Model보다 오히려 정확도가 떨어진다.

$$\text{Accuracy} \iff \# \text{ of Measurement(Computation)}$$

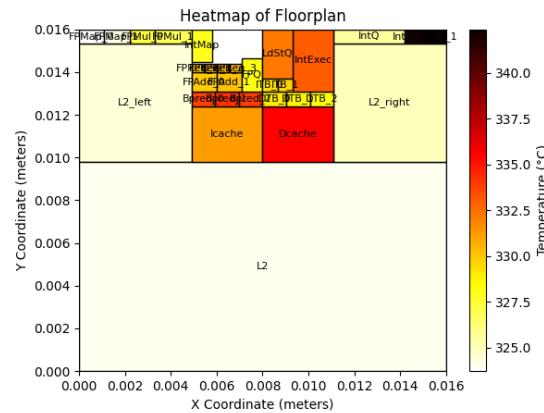
따라서 이들을 적절히 활용하기 위해서는 어떤 환경 인지가 중요하다. 온도 변화에 빠른 대응이 필요한 환경에서는 Block Model이나 낮은 granularity의 Grid Model이 적절하고, 정확한 온도 측정을 요구하는 환경에서는 높은 granularity의 Grid Model이 적절하다.

3. Advance Block Model

온도 측정 결과 `IntReg`에서 가장 많은 열을 내고 있어서 이들을 가장자리로 치우면 중간에 모여있는 part들의 온도가 내려가지 않을까 하는 가설에서 floorplan을 중앙에 집중되어있는 part들을 가장자리로 조금 치워주었다.



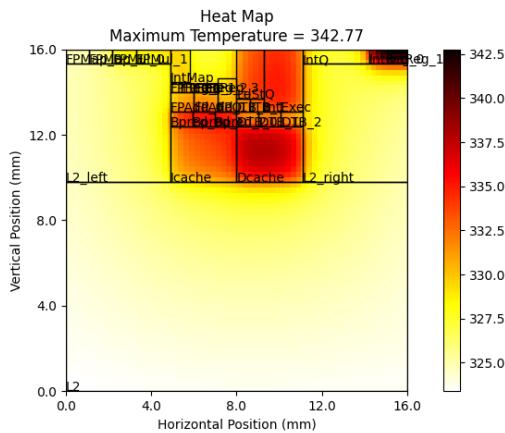
Heatmap for ev6.flp



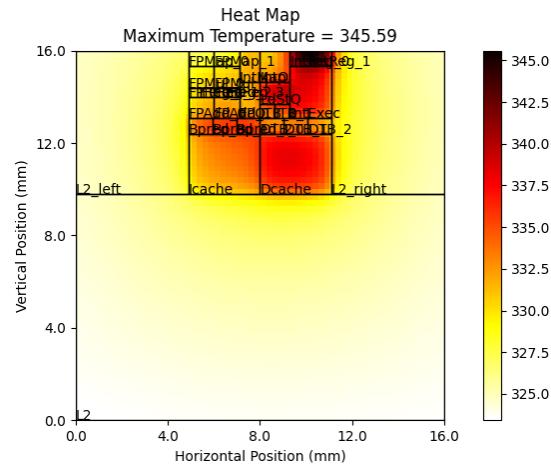
Heatmap for my.flp

이와 같이 `IntReg_0` 와 `IntReg_1` 을 `L2_right` 위로 배치한 결과, `IntExec` 과 `LdStQ` 의 온도가 내려간 것을 볼 수 있다. 이는 뜨거운 컴포넌트의 주위에 위치한 컴포넌트도 해당 컴포넌트의 영향을 받는다는 것을 보여준다. 또한 `L2_left` 위에 배치한 `FPMap` 의 경우 이전 보다 온도가 훨씬 떨어진 것을 확인할 수 있다.

64X64 Grid Model로 측정한 두 flooplanning의 온도 차이를 보면, 실제로 뜨거운 컴포넌트가 주변 컴포넌트에 영향을 미치는 것을 줄인 것을 확인할 수 있다. 또한 측정된 최대 온도도 `my.flp` 가 더 낮았다.



my.flp 64X64 grid model heatmap



Heatmap for 64X64 Grid Model

다음으로 64X64 Grid Model과의 transient temperature 차이를 구했다. Grid Model은 Block Model보다 더 정확하기 때문에 `my.flp` 의 Grid와 Block의 MSE를 구해서 원래의

`ev6.flp` 의 MSE와 비교하여 성능이 더 좋아졌는지 확인해볼 수 있다.

	Part	MSE
0	L2_left	2.003465
1	L2	0.058679
2	L2_right	1.236391
3	Icache	5.089662
4	Dcache	6.264734
5	Bpred_0	26.010934
6	Bpred_1	22.896657
7	Bpred_2	20.959944
8	DTB_0	10.877546
9	DTB_1	8.456335
10	DTB_2	4.576530
11	FPAdd_0	25.417614
12	FPAdd_1	32.990177
13	FPReg_0	42.550001
14	FPReg_1	50.806156
15	FPReg_2	65.530685
16	FPReg_3	68.471568
17	FPMul_0	6.116957
18	FPMul_1	14.594335
19	FPMul_0	0.780146
20	FPMul_1	1.383145
21	IntMap	43.332357
22	IntQ	2.692335
23	IntReg_0	32.197294
24	IntReg_1	10.430352
25	IntExec	9.213289
26	FPQ	36.603048
27	LdStQ	28.106266
28	ITB_0	17.853058
29	ITB_1	11.822601

MSE for my.flp

	Part	MSE
0	L2_left	2.598825
1	L2	1.162221
2	L2_right	4.409828
3	Icache	3.304618
4	Dcache	0.322364
5	Bpred_0	0.144805
6	Bpred_1	1.875598
7	Bpred_2	2.848895
8	DTB_0	22.562237
9	DTB_1	26.703440
10	DTB_2	17.366736
11	FPAdd_0	3.988266
12	FPAdd_1	9.778494
13	FPReg_0	0.711721
14	FPReg_1	3.125051
15	FPReg_2	4.851282
16	FPReg_3	4.744022
17	FPMul_0	1.929007
18	FPMul_1	5.432403
19	FPMul_0	4.422862
20	FPMul_1	8.986865
21	IntMap	5.160999
22	IntQ	31.676104
23	IntReg_0	0.116016
24	IntReg_1	1.807917
25	IntExec	4.062680
26	FPQ	16.850814
27	LdStQ	0.851369
28	ITB_0	14.919511
29	ITB_1	17.803345

MSE for ev6.flp

먼저, 왼쪽은 새로 만든 `my.flp` 의 Block Model과 64X64 Grid Model의 part 별로 MSE를 구한 값이다. 그리고 오른쪽은 `ev6.flp` 의 Block Model과 64X64 Grid Model의 part 별 MSE 값이다. 각 컴포넌트 별 MSE를 보면, 크기가 큰 컴포넌트들은 `my.flp` 의 MSE 값이 더 낮고, 컴포넌트의 크기가 작아질수록 `ev6.flp` 의 MSE 값이 더 낮게 나온다. 이러한 이유에 대해 생각을 해보면, 각 컴포넌트의 실제 온도를 낮추기 위해 중앙에 있는 작은 컴포넌트들을 가장자리로 배치시켰기 때문에, 이들의 주변부 온도가 `ev6.flp` 에 비해 낮게 측정이 될 것인데, `ev6.flp` 에서는 오히려 주변의 뜨거운 컴포넌트들 때문에 중심부 온도와 주변부 온도가 비교적

고르게 분포되었던 것 같다. 하지만 my.flp에서는 주변부 온도가 낮기 때문에 컴포넌트의 온도를 정확히 측정되지 못했던 것 같다. 이로써 알게된 사실로는 온도를 낮추기 위한 floorplanning을 하는 것도 중요하지만 측정값의 정확도도 floorplanning의 영향을 많이 받음을 알 수 있다. 따라서 작은 컴포넌트들이 중앙에 밀집된 형태가 컴포넌트의 온도를 높일 수 있지만 Block Model에서 비교적 빠르고 정확한 온도 측정을 달성할 수 있게 되는 trade-off가 있다.

$$Temperature Decrease \iff Accuracy on Block Modeling$$