Soham Gadgil GTID: 903130189

Project 2 ECE 4100 Experiments:

Experiment 1:

a) gcc.100k.trace

The following table shows the different values tested for F, k0, k1, k2.

ROB (r)	Dispatch Rate (f)	k0 (j)	k1 (k)	k2 (l)	PREGs (p)	IPC
(Default) 12	4	3	2	1	32	2.933067
12	3	3	2	1	32	2.727397
10	4	3	1	1	32	2.412429
10	4	2	2	1	32	2.614516

Insights:

For this trace, multiple instructions (close to 3) get executed and/or retired in one cycle. Thus, as it can be seen above, reducing the number of function units causes a big change in the IPC since the instructions must wait to get scheduled because of limited functional units. Even reducing the fetch rate has a significant impact on IPC but it is not as big as the impact due to change in functional units.

Thus, there are no parameters that can be further reduced to get 98% of the default IPC. The closest we can get is by reducing the fetch rate to 3, but that is still 93% of the default IPC. This is the closest we can get to the default IPC.

Thus, the final configuration is ROB – 12, Dispatch Rate – 3, k0 units – 3, k1 units – 2, k2 units – 1, PREGs – 32, IPC – 2.727397, % of default IPC – 93%

b) gobmk.100k.trace

The following table shows the different values tested for F, k0, k1, k2.

ROB (r)	Dispatch Rate (f)	k0 (j)	k1 (k)	k2 (l)	PREGs (p)	IPC
(Default) 12	4	3	2	1	32	1.600487
12	3	3	2	1	32	1.572401
10	3	2	2	1	32	1.569243

Insights:

For this trace, only 1 or 2 instructions are executed or retired in each cycle, because of which reducing the dispatch rate or the number of functional units does not affect the IPC much as long as it is not too low, as it can be seen above.

In the end, by reducing the dispatch rate and k0 functional units, we are still able to get an IPC which is 98.05% of the default IPC.

Thus, the final configuration is ROB – 10, Dispatch Rate – 3, k0 units – 2, k1 units – 2, k2 units – 1, PREGs – 32, IPC – 1.569243, % of default IPC – 98.05%

c) hmmer.100k.trace

The following table shows the different values tested for F, k0, k1, k2.

ROB (r)	Dispatch Rate (f)	k0 (j)	k1 (k)	k2 (l)	PREGs (p)	IPC
(Default) 12	4	3	2	1	32	2.756263
10	4	2	2	1	32	2.614516
10	3	2	2	1	32	2.517433
12	3	3	2	1	32	2.615405

Insights:

For this trace, there are multiple instructions (2 – 3) getting executed and/or retired in the same cycle. As a result the fetch rate is very sensitive to reduction. If the fetch rate is reduced even by 1, it caused a significant drop in IPC. Also, reducing the functional units cause the IPC to drop by a lot since there are instructions stalling to be executed due to limited functional units.

As a result, there are no parameters that can be further reduced to get 98% of the default IPC. The closest we can get is by reducing the fetch rate to 3, but that is still 95% of the default IPC. This is the closest we can get to the default IPC.

Thus, the final configuration is ROB – 12, Dispatch Rate – 3, k0 units – 3, k1 units – 2, k2 units – 1, PREGs – 32, IPC – 2.615405, % of default IPC – 95%

d) mcf.100k.trace

The following table shows the different values tested for F, k0, k1, k2.

ROB (r)	Dispatch Rate (f)	k0 (j)	k1 (k)	k2 (l)	PREGs (p)	IPC
(Default) 12	4	3	2	1	32	0.590898
12	3	3	2	1	32	0.581172
10	3	2	2	1	32	0.580511
8	3	1	2	1	32	0.567012

Insights:

For this trace, there is barely 1 instruction getting executed or retired in one cycle. Thus, there are not many instructions waiting to get executed or completed. As a result reducing the fetch rate and the functional units does not have a big impact on the IPC and we can get very close to the default IPC with these reductions.

In the end, with reductions in both fetch rate and functional units, we are still able to get an IPC which is 98.24% of the default IPC.

Thus, the final configuration is ROB – 10, Dispatch Rate – 3, k0 units – 2, k1 units – 2, k2 units – 1, PREGs – 32, IPC – 0.580511, % of default IPC – 98.24%

Experiment 2:

a) gcc.100k.trace

The following table shows the different values tested for the ROB entries:

ROB (r)	Dispatch Rate (f)	k0 (j)	k1 (k)	k2 (l)	PREGs (p)	IPC
(Default) 12	4	3	2	1	32	2.933067
11	4	3	2	1	32	2.791269

Insights:

Since for this trace there are multiple instructions getting executed and/or retired in each cycle on average, reducing the number of ROB entries has a significant impact on IPC. This might be because the ROB gets filled up because of less space which causes more instructions to stall.

In the end, there are the ROB entries cannot be further reduced to get 98% of the default IPC. The closest we can get is with 11 ROB entries, which is only about 95.16% of the default IPC.

Thus, the final configuration is ROB – 11, Dispatch Rate – 4, k0 units – 3, k1 units – 2, k2 units – 1, PREGs – 32, IPC – 2.791269, % of default IPC – 95.16%

b) gobmk.100k.trace

The following table shows the different values tested for the ROB entries:

ROB (r)	Dispatch Rate (f)	k0 (j)	k1 (k)	k2 (l)	PREGs (p)	IPC
(Default) 12	4	3	2	1	32	1.600487
11	4	3	2	1	32	1.598644
10	4	3	2	1	32	1.595913
8	4	3	2	1	32	1.573787
7	4	3	2	1	32	1.546934

Insights:

Since for this trace, there are only a few instructions getting retired/executed per cycle, the ROB never really gets full to cause too many instructions to stall. As can be seen from the results above, reducing the ROB entries significantly does not impact the IPC much. As a result, we are easily able to achieve more than 98% of the default IPC with less ROB entries.

In the end, reducing the ROB entries to 8 still gives us an IPC which is 98.33% of the default IPC.

Thus, the final configuration is ROB – 8, Dispatch Rate – 4, k0 units – 3, k1 units – 2, k2 units – 1, PREGs – 32, IPC – 1.573787, % of default IPC – 98.33%

c) hmmer.100k.trace

The following table shows the different values tested for the ROB entries:

ROB (r)	Dispatch Rate (f)	k0 (j)	k1 (k)	k2 (l)	PREGs (p)	IPC
(Default) 12	4	3	2	1	32	2.756263
11	4	3	2	1	32	2.720792
10	4	3	2	1	32	2.595178

Insights:

Since for this trace, there are many instructions either completing or retiring in each cycle, the ROB is kept pretty busy most of the time and once the ROB gets full, instructions start to stall, reducing the IPC. Thus, there is not much change that can be done the ROB entries as a large reduction has a big impact on IPC.

In the end, we can reduce the ROB entries to 11 to get an IPC that is 98.71% of the default IPC.

Thus, the final configuration is ROB – 11, Dispatch Rate – 4, k0 units – 3, k1 units – 2, k2 units – 1, PREGs – 32, IPC – 2.720792, % of default IPC – 98.71%

d) mcf.100k.trace

The following table shows the different values tested for the ROB entries:

ROB (r)	Dispatch Rate (f)	k0 (j)	k1 (k)	k2 (l)	PREGs (p)	IPC
(Default) 12	4	3	2	1	32	0.590898
10	4	3	2	1	32	0.59047
9	4	3	2	1	32	0.59225
8	4	3	2	1	32	0.590127
6	4	3	2	1	32	0.587013
5	4	3	2	1	32	0.57971
4	4	3	2	1	32	0.568919

Insights:

Since for this trace, there is barely one instruction getting executed or retired in each cycle, the ROB rarely gets full or causes any stalls. Most of the time, the ROB has empty slots available to dispatch instructions. Thus, reducing the ROB entries to more than half of the default doesn't have much impact on the IPC, making 98% attainable easily.

In the end, we can reduce the ROB entries to 5 to get an IPC that is 98.1% of the default IPC.

Thus, the final configuration is ROB – 5, Dispatch Rate – 4, k0 units – 3, k1 units – 2, k2 units – 1, PREGs – 32, IPC – 0.57971, % of default IPC – 98.1%