Name: Pritam Bhattacharyya

GTID: 902989158

Email: pritam3@gatech.edu

Experiment Objective:

The objective of the experiment was to determine the minimum hardware, required to simulate an out-of-order superscalar processor, which can provide nearly (>95%) of the highest IPC value. The operation of the processor was based on Tomasulo algorithm and was subject to the following constraints:

1. The maximum number of FUs (k0, k1 and k2) that we can consider in our experiments are 2.
2. The fetch rate can be either 4 or 8 instructions/cycle.
3. The latency for every FU is 1 clock cycle.
4. The dispatch queue can be infinitely big.
5. The maximum number of instructions that can exist in the schedule queue at any point in time is given by:

Schedule Queue size = No. of FU0 + No. of FU1 + No. of FU2

Experiment Method:

The experiment was conducted on each trace individually by varying the relevant parameters, while keeping the other parameters constant. The trend found from such variations were recorded. Then, based on the observations of those trends, the optimal hardware was accordingly derived and verified.

The parameters that were user defined were:

R = Number of result buses

f = Number of instructions to be fetched per cycle

k0 = Number of FU0s

k1 = Number of FU1s

K2 = Number of FU2s

Trends observed for "gcc.100k.trace" trace file

Variation of IPC with varying R (keeping other values constant at their possible maximum)

R	IPC
1	0.999940
2	1.925336
3	2.366752
4	2.411091
5	2.422070
6	2.422070
7	2.422070

[**NOTE:** for this trend, the values of other variables are: f = 8, k0 = 2, k1 = 2, k2 = 2]

As seen from the trend, the IPC saturates after R increases beyond 5. The reason for this is easily understood if we consider the fact that there are 6 total FUs in this simulation and beyond 6 result buses, it is equivalent to every FU having their individual result bus, which essentially saturates the amount of speedup that can be achieved. Increasing the count of beyond 6 makes no impact as in this case, the extra result buses will always stay unused as there will not be any FU to write data to those result buses.

Thus, the highest IPC that can be achieved in 'gcc.100k.trace' is 2.422070 (as all the other parameters are maxed out for the above simulation). Which implies that we have to find a hardware configuration to achieve a minimum IPC value of 2.3009665.

Variation of IPC with varying k0 (keeping other values constant at their possible maximum)

k0	IPC
1	1.727981
2	2.422070

[**NOTE:** for this trend, the values of other variables are: R = 6, f = 8, k1 = 2, k2 = 2]

As we can see, when we decrease the number of FU0, the IPC decreases which is expected as the processor has lesser number of functional unit to process the same amount of workload. Thus, the IPC decreases.

However, as we can see that even with the other variables maxed out, the IPC decreases below 95% of the maximum possible IPC. Therefore, decreasing the number of FU0 below 2 is not an option.

Variation of IPC with varying k1 (keeping other values constant at their possible maximum)

k1	IPC
1	2.024947
2	2.422070

[**NOTE:** for this trend, the values of other variables are: R = 6, f = 8, k0 = 2, k2 = 2]

As we can see, when we decrease the number of FU1, the IPC decreases which is expected as the processor has lesser number of functional unit to process the same amount of workload. Thus, the IPC decreases.

However, as we can see that even with the other variables maxed out, the IPC decreases below 95% of the maximum possible IPC. Therefore, decreasing the number of FU1 below 2 is not an option.

Variation of IPC with varying k2 (keeping other values constant at their possible maximum)

k2	IPC
1	2.072453
2	2.422070

[**NOTE:** for this trend, the values of other variables are: R = 6, f = 8, k0 = 2, k1 = 2]

As we can see, when we decrease the number of FU2, the IPC decreases which is expected as the processor has lesser number of functional unit to process the same amount of workload. Thus, the IPC decreases.

However, as we can see that even with the other variables maxed out, the IPC decreases below 95% of the maximum possible IPC. Therefore, decreasing the number of FU2 below 2 is not an option.

Variation of IPC with varying f (keeping other values constant at their possible maximum)

f	IPC
4	2.422070
8	2.422070

[**NOTE:** for this trend, the values of other variables are: R = 6, k0 = 2, k1 = 2, k2 = 2]

As we can see, varying the f does not affect the IPC because even though initially fetching of more data in a cycle may help in scheduling more instructions in the beginning of the trace. However, as the trace runs on and the schedule queue fills up, the execution of data no longer depends on the amount of data fetched (as the dispatch queue is infinite in size). Rather, now the instruction retiring depends on the rate at which instructions are executed and removed from the schedule queue.

Thus, we can decrease the fetch rate in our case without significant compromise in terms of IPC.

Optimal Solution

Thus, based on the above trends, we can say that the optimal hardware condition will be achieved when the parameters are as follows:

R = 3; f = 4; k0 = 2; k1 = 2; k3 = 2

The optimal IPC for this configuration is = **2.366752**

Trends observed for "gobmk.100k.trace" trace file

Variation of IPC with varying R (keeping other values constant at their possible maximum)

R	IPC
1	0.999830
2	1.881857
3	2.304625
4	2.361721
5	2.364457
6	2.364457
7	2.364457

[**NOTE:** for this trend, the values of other variables are: f = 8, k0 = 2, k1 = 2, k2 = 2]

As seen from the trend, the IPC saturates after R increases beyond 5. The reason for this is easily understood if we consider the fact that there are 6 total FUs in this simulation and beyond 6 result buses, it is equivalent to every FU having their individual result bus, which essentially saturates the amount of speedup that can be achieved. Increasing the count of beyond 6 makes no impact as in this case, the extra result buses will always stay unused as there will not be any FU to write data to those result buses.

Thus, the highest IPC that can be achieved in 'gobmk.100k.trace' is 2.364457 (as all the other parameters are maxed out for the above simulation). Which implies that we have to find a hardware configuration to achieve a minimum IPC value of 2.24623415.

Variation of IPC with varying k0 (keeping other values constant at their possible maximum)

k0	IPC
1	1.868635
2	2.364457

[**NOTE:** for this trend, the values of other variables are: R = 6, f = 8, k1 = 2, k2 = 2]

As we can see, when we decrease the number of FU0, the IPC decreases which is expected as the processor has lesser number of functional unit to process the same amount of workload. Thus, the IPC decreases.

However, as we can see that even with the other variables maxed out, the IPC decreases below 95% of the maximum possible IPC. Therefore, decreasing the number of FU0 below 2 is not an option.

Variation of IPC with varying k1 (keeping other values constant at their possible maximum)

k1	IPC
1	2.039235
2	2.364457

[**NOTE:** for this trend, the values of other variables are: R = 6, f = 8, k0 = 2, k2 = 2]

As we can see, when we decrease the number of FU1, the IPC decreases which is expected as the processor has lesser number of functional unit to process the same amount of workload. Thus, the IPC decreases.

However, as we can see that even with the other variables maxed out, the IPC decreases below 95% of the maximum possible IPC. Therefore, decreasing the number of FU1 below 2 is not an option.

Variation of IPC with varying k2 (keeping other values constant at their possible maximum)

k2	IPC
1	2.030663
2	2.364457

[**NOTE:** for this trend, the values of other variables are: R = 6, f = 8, k0 = 2, k1 = 2]

As we can see, when we decrease the number of FU2, the IPC decreases which is expected as the processor has lesser number of functional unit to process the same amount of workload. Thus, the IPC decreases.

However, as we can see that even with the other variables maxed out, the IPC decreases below 95% of the maximum possible IPC. Therefore, decreasing the number of FU2 below 2 is not an option.

Variation of IPC with varying f (keeping other values constant at their possible maximum)

f	IPC
4	2.364457
8	2.364457

[**NOTE:** for this trend, the values of other variables are: R = 6, k0 = 2, k1 = 2, k2 = 2]

As we can see, varying the f does not affect the IPC because even though initially fetching of more data in a cycle may help in scheduling more instructions in the beginning of the trace. However, as the trace runs on and the schedule queue fills up, the execution of data no longer depends on the amount of data fetched (as the dispatch queue is infinite in size). Rather, now the instruction retiring depends on the rate at which instructions are executed and removed from the schedule queue.

Thus, we can decrease the fetch rate in our case without significant compromise in terms of IPC.

Optimal Solution

Thus, based on the above trends, we can say that the optimal hardware condition will be achieved when the parameters are as follows:

R = 3; f = 4; k0 = 2; k1 = 2; k3 = 2

The optimal IPC for this configuration is = **2.304625**

Trends observed for "hmmer.100k.trace" trace file

Variation of IPC with varying R (keeping other values constant at their possible maximum)

R	IPC
1	0.999960
2	1.834055
3	2.206385
4	2.262546
5	2.266906
6	2.266854
7	2.266854

[**NOTE:** for this trend, the values of other variables are: f = 8, k0 = 2, k1 = 2, k2 = 2]

As seen from the trend, the IPC saturates after R increases beyond 5. The reason for this is easily understood if we consider the fact that there are 6 total FUs in this simulation and beyond 6 result buses, it is equivalent to every FU having their individual result bus, which essentially saturates the amount of speedup that can be achieved. Increasing the count of beyond 6 makes no impact as in this case, the extra result buses will always stay unused as there will not be any FU to write data to those result buses.

Thus, the highest IPC that can be achieved in 'hmmer.100k.trace' is 2.266906 (as all the other parameters are maxed out for the above simulation). Which implies that we have to find a hardware configuration to achieve a minimum IPC value of 2.1535607.

Variation of IPC with varying k0 (keeping other values constant at their possible maximum)

k0	IPC
1	1.702302
2	2.266854

[**NOTE:** for this trend, the values of other variables are: R = 6, f = 8, k1 = 2, k2 = 2]

As we can see, when we decrease the number of FU0, the IPC decreases which is expected as the processor has lesser number of functional unit to process the same amount of workload. Thus, the IPC decreases.

However, as we can see that even with the other variables maxed out, the IPC decreases below 95% of the maximum possible IPC. Therefore, decreasing the number of FU0 below 2 is not an option.

Variation of IPC with varying k1 (keeping other values constant at their possible maximum)

k1	IPC
1	1.879947
2	2.266854

[**NOTE:** for this trend, the values of other variables are: R = 6, f = 8, k0 = 2, k2 = 2]

As we can see, when we decrease the number of FU1, the IPC decreases which is expected as the processor has lesser number of functional unit to process the same amount of workload. Thus, the IPC decreases.

However, as we can see that even with the other variables maxed out, the IPC decreases below 95% of the maximum possible IPC. Therefore, decreasing the number of FU1 below 2 is not an option.

Variation of IPC with varying k2 (keeping other values constant at their possible maximum)

k2	IPC
1	1.943181
2	2.266854

[**NOTE:** for this trend, the values of other variables are: R = 6, f = 8, k0 = 2, k1 = 2]

As we can see, when we decrease the number of FU2, the IPC decreases which is expected as the processor has lesser number of functional unit to process the same amount of workload. Thus, the IPC decreases.

However, as we can see that even with the other variables maxed out, the IPC decreases below 95% of the maximum possible IPC. Therefore, decreasing the number of FU2 below 2 is not an option.

Variation of IPC with varying f (keeping other values constant at their possible maximum)

f	IPC
4	2.266854
8	2.266854

[**NOTE:** for this trend, the values of other variables are: R = 6, k0 = 2, k1 = 2, k2 = 2]

As we can see, varying the f does not affect the IPC because even though initially fetching of more data in a cycle may help in scheduling more instructions in the beginning of the trace. However, as the trace runs on and the schedule queue fills up, the execution of data no longer depends on the amount of data fetched (as the dispatch queue is infinite in size). Rather, now the instruction retiring depends on the rate at which instructions are executed and removed from the schedule queue.

Thus, we can decrease the fetch rate in our case without significant compromise in terms of IPC.

Optimal Solution

Thus, based on the above trends, we can say that the optimal hardware condition will be achieved when the parameters are as follows:

R = 3; f = 4; k0 = 2; k1 = 2; k3 = 2

The optimal IPC for this configuration is = **2.206385**

Trends observed for "mcf.100k.trace" trace file

Variation of IPC with varying R (keeping other values constant at their possible maximum)

R	IPC
1	0.998582
2	1.878887
3	2.324554
4	2.369444
5	2.369444
6	2.369444
7	2.396444

[**NOTE:** for this trend, the values of other variables are: f = 8, k0 = 2, k1 = 2, k2 = 2]

As seen from the trend, the IPC saturates after R increases beyond 5. The reason for this is easily understood if we consider the fact that there are 6 total FUs in this simulation and beyond 6 result buses, it is equivalent to every FU having their individual result bus, which essentially saturates the amount of speedup that can be achieved. Increasing the count of beyond 6 makes no impact as in this case, the extra result buses will always stay unused as there will not be any FU to write data to those result buses.

Thus, the highest IPC that can be achieved in 'mcf.100k.trace' is 2.369444 (as all the other parameters are maxed out for the above simulation). Which implies that we have to find a hardware configuration to achieve a minimum IPC value of 2.2509718.

Variation of IPC with varying k0 (keeping other values constant at their possible maximum)

k0	IPC
1	1.857079
2	2.396444

[**NOTE:** for this trend, the values of other variables are: R = 6, f = 8, k1 = 2, k2 = 2]

As we can see, when we decrease the number of FU0, the IPC decreases which is expected as the processor has lesser number of functional unit to process the same amount of workload. Thus, the IPC decreases.

However, as we can see that even with the other variables maxed out, the IPC decreases below 95% of the maximum possible IPC. Therefore, decreasing the number of FU0 below 2 is not an option.

Variation of IPC with varying k1 (keeping other values constant at their possible maximum)

k1	IPC
1	1.897785
2	2.396444

[**NOTE:** for this trend, the values of other variables are: R = 6, f = 8, k0 = 2, k2 = 2]

As we can see, when we decrease the number of FU1, the IPC decreases which is expected as the processor has lesser number of functional unit to process the same amount of workload. Thus, the IPC decreases.

However, as we can see that even with the other variables maxed out, the IPC decreases below 95% of the maximum possible IPC. Therefore, decreasing the number of FU1 below 2 is not an option.

Variation of IPC with varying k2 (keeping other values constant at their possible maximum)

k2	IPC
1	1.989496
2	2.396444

[**NOTE:** for this trend, the values of other variables are: R = 6, f = 8, k0 = 2, k1 = 2]

As we can see, when we decrease the number of FU2, the IPC decreases which is expected as the processor has lesser number of functional unit to process the same amount of workload. Thus, the IPC decreases.

However, as we can see that even with the other variables maxed out, the IPC decreases below 95% of the maximum possible IPC. Therefore, decreasing the number of FU2 below 2 is not an option.

Variation of IPC with varying f (keeping other values constant at their possible maximum)

f	IPC
4	2.396444
8	2.396444

[**NOTE:** for this trend, the values of other variables are: R = 6, k0 = 2, k1 = 2, k2 = 2]

As we can see, varying the f does not affect the IPC because even though initially fetching of more data in a cycle may help in scheduling more instructions in the beginning of the trace. However, as the trace runs on and the schedule queue fills up, the execution of data no longer depends on the amount of data fetched (as the dispatch queue is infinite in size). Rather, now the instruction retiring depends on the rate at which instructions are executed and removed from the schedule queue.

Thus, we can decrease the fetch rate in our case without significant compromise in terms of IPC.

Optimal Solution

Thus, based on the above trends, we can say that the optimal hardware condition will be achieved when the parameters are as follows:

R = 3; f = 4; k0 = 2; k1 = 2; k3 = 2

The optimal IPC for this configuration is = **2.324500**