**NC State University**

**Department of Electrical and Computer Engineering**

**ECE 463/521: Fall 2015 (Rotenberg)**

# Project #3: Dynamic Instruction Scheduling

**by**

**Laxman Sole**

NCSU Honor Pledge: "I have neither given nor received unauthorized aid on this test or assignment."

Student's electronic signature: Laxman Sole

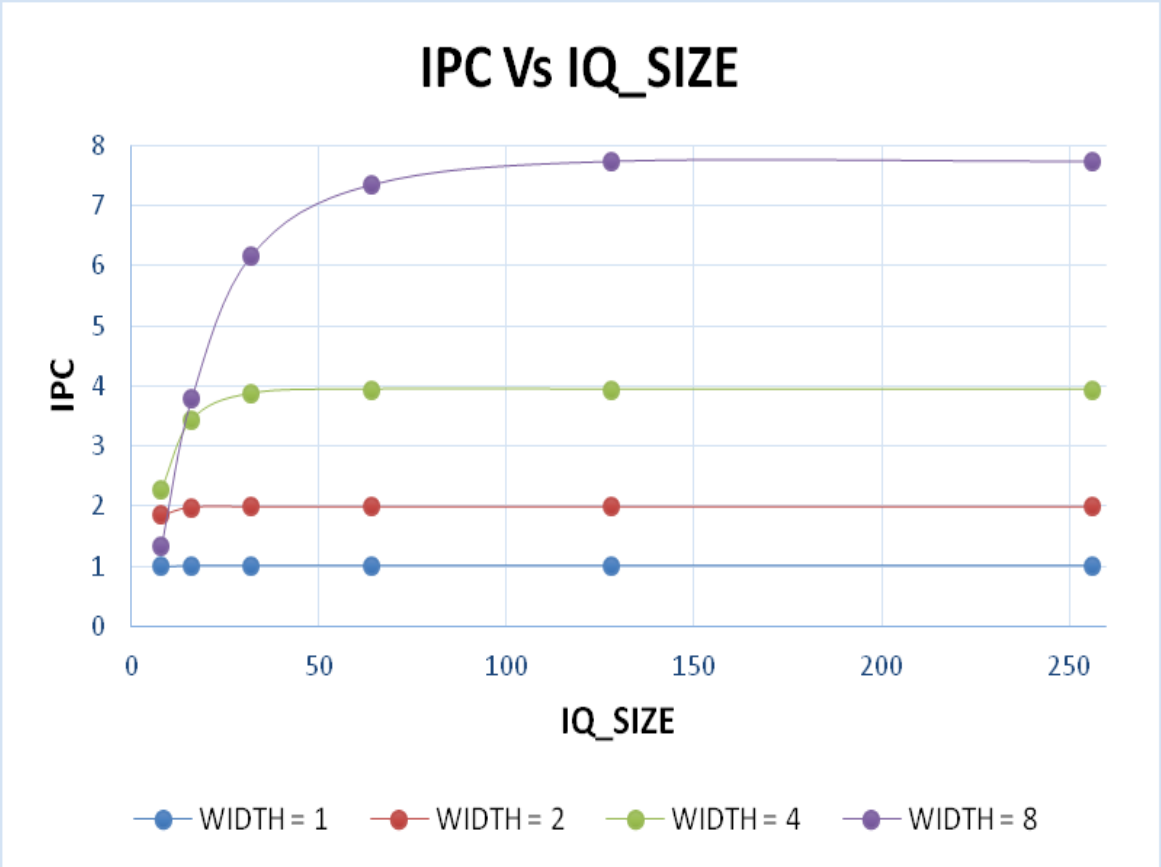
Course number: 521

# Large ROB, effect of IQ_SIZE

## Graph*:* GCC Trace

**Graph of IPC vs IQ_SIZE with ROB_SIZE = 512 for each WIDTH:**

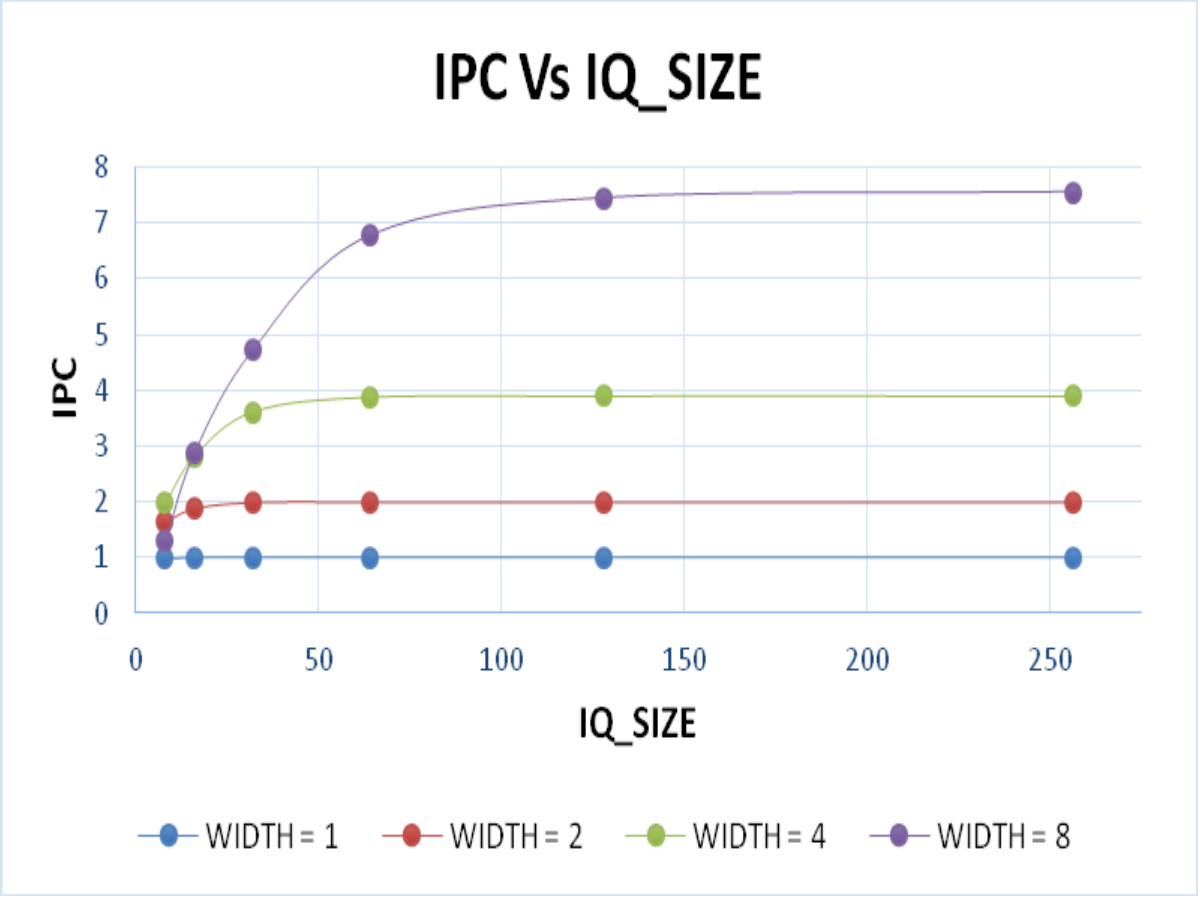
| IQ_SIZE | WIDTH-1 | WIDTH-2 | WIDTH-4 | WIDTH-8 |
|---|---|---|---|---|
| 8 | 0.99 | 1.85 | 2.28 | 1.33 |
| 16 | 1 | 1.98 | 3.43 | 3.78 |
| 32 | 1 | 1.99 | 3.87 | 6.17 |
| 64 | 1 | 1.99 | 3.94 | 7.34 |
| 128 | 1 | 1.99 | 3.94 | 7.73 |
| 256 | 1 | 1.99 | 3.94 | 7.73 |



## Graph*:* Perl Trace

**Graph of IPC vs IQ_SIZE with ROB_SIZE = 512 for each WIDTH:**

| IQ_SIZE | WIDTH-1 | WIDTH-2 | WIDTH-4 | WIDTH-8 |
|---|---|---|---|---|
| 8 | 0.98 | 1.64 | 1.98 | 1.29 |
| 16 | 1 | 1.87 | 2.82 | 2.88 |
| 32 | 1 | 1.97 | 3.61 | 4.72 |
| 64 | 1 | 1.98 | 3.87 | 6.77 |
| 128 | 1 | 1.98 | 3.89 | 7.44 |
| 256 | 1 | 1.98 | 3.89 | 7.55 |



# Graph Analysis

## GCC Trace

| WIDTH | IPC for IQ_SIZE = 256 | within %5 of IPC for largest IQ_SIZE | Optimized IQ_SIZE |
|---|---|---|---|
| 1 | 1 | 0.95 | 8 |
| 2 | 1.99 | 1.8905 | 16 |
| 4 | 3.94 | 3.743 | 64 |
| 8 | 7.73 | 7.3435 | 128 |

## Perl Trace

| WIDTH | IPC for IQ_SIZE = 256 | within %5 of IPC for largest IQ_SIZE | Optimized IQ_SIZE |
|---|---|---|---|
| 1 | 1 | 0.95 | 8 |
| 2 | 1.98 | 1.881 | 32 |
| 4 | 3.89 | 3.6955 | 64 |
| 8 | 7.55 | 7.1725 | 128 |

## Discussion

a)

For a given pipeline, if ROB is not a bottleneck i.e. ROB is sufficiently large, then IQ_SiZE can become the bottleneck. From the results of GCC and Perl traces, we can see that performance (IPC) increases with increased size of Issue queue for same WIDTH. Performance reaches to top at particular IQ_SIZE.

This happens because if there are more no f dependent instruction is pipeline then IQ is the only storage for the instructions which are not ready and more the storage, more time we can fetch new instructions without stalling pipeline. This means if we have large IQ, will never stall a pipeline and we will get the theoretically possible IPC of of 1. However, after sufficient size, increasing size furthermore is not effective as we have already achieved the peak IPC.

In general case, from the graph data, we can see that, we can get the highest performance for IQ_SIZE which is 8-16 times the WIDTH.

b)

GCC benchmark shows slightly higher IPC than perl for the same microarchitecture configuration. Though the microarchitecture configuration is same, IPC is also depends on how instructions are scheduled by the compiler i.e. static scheduling. If there are more independent instructions available then pipeline performs smoothly giving the best performance. If there are many dependent instructions close to each other, then data hazards are bound to happen and to avoid those hazards, in some cases pipeline has to stall. This degrades the IPC.

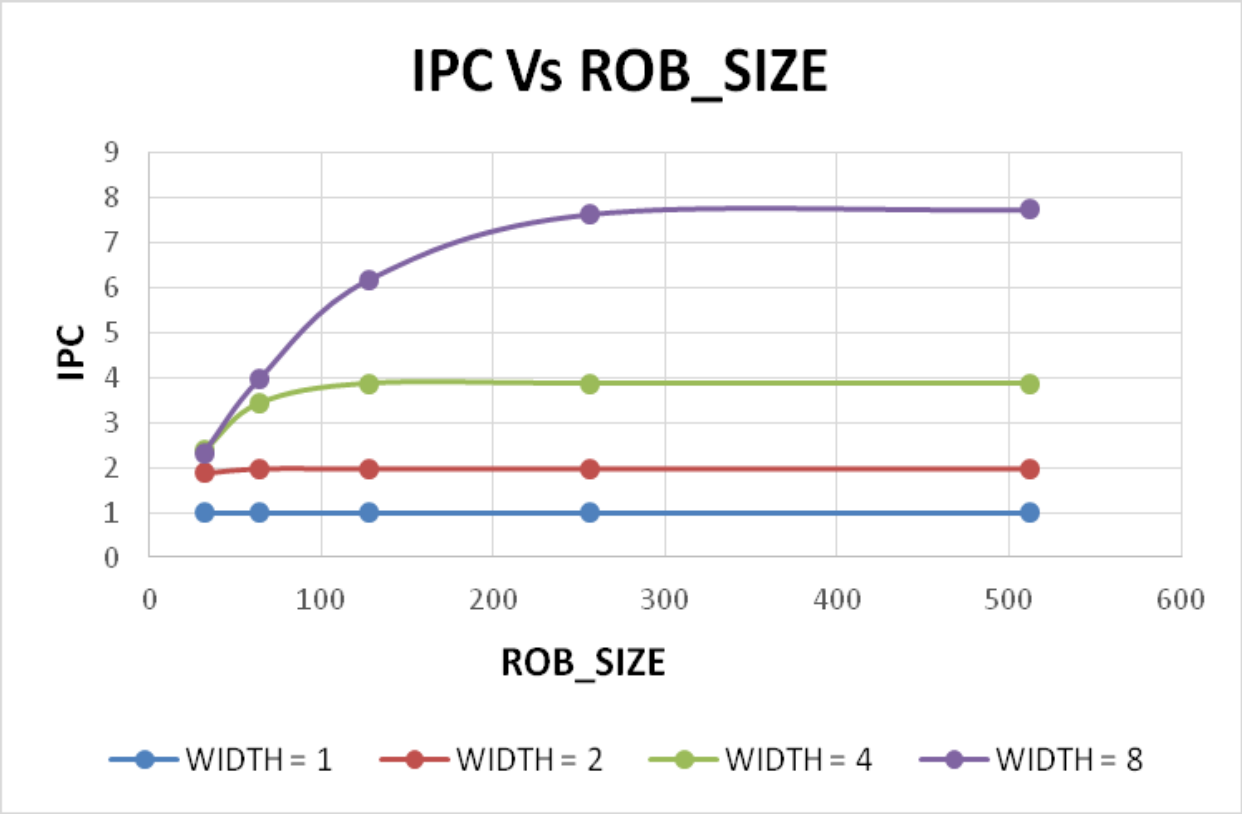
So performance is mainly depending upon microarchitecture configuration and nature of the program. Microarchitecture tries to give best performance but if there are not many opportunities for instruction level parallelism, then it results in lower IPC than expected.

# Effect of ROB_SIZE

## Graph: GCC Trace

### Graph of IPC Vs ROB_SIZE with optimized IQ_SIZE for each WIDTH:

| ROB_SIZE | WIDTH = 1, IQ_SIZE = 8 | WIDTH = 2, IQ_SIZE = 16 | WIDTH = 4, IQ_SIZE = 64 | WIDTH = 8, IQ_SIZE = 128 |
|---|---|---|---|---|
| 32 | 0.98 | 1.79 | 2.19 | 2.18 |
| 64 | 0.98 | 1.97 | 3.16 | 3.51 |
| 128 | 0.98 | 1.97 | 3.83 | 5.36 |
| 256 | 0.98 | 1.97 | 3.87 | 7.05 |
| 512 | 0.98 | 1.97 | 3.87 | 7.44 |



## Graph: Perl Trace

### Graph of IPC vs ROB_SIZE with optimized IQ_SIZE for each WIDTH:

| ROB_SIZE | WIDTH = 1, IQ_SIZE = 8 | WIDTH = 2, IQ_SIZE = 16 | WIDTH = 1, IQ_SIZE = 64 | WIDTH = 1, IQ_SIZE = 128 |
|---|---|---|---|---|
| 32 | 0.98 | 1.79 | 2.19 | 2.18 |
| 64 | 0.98 | 1.97 | 3.16 | 3.51 |
| 128 | 0.98 | 1.97 | 3.83 | 5.36 |
| 256 | 0.98 | 1.97 | 3.87 | 7.05 |
| 512 | 0.98 | 1.97 | 3.87 | 7.44 |