

Latency vs. # Tokens

As defined in chapter 4 page 2:

Latency is the time between tokens consumed at inputs and generated outputs

Whenever the pipeline is empty (always correct for the 1st token or when the token period (spread) is wider than $\max\{FL + BL\}$), latency is the function of (1) the number of buffers and (2) the forward latency FL and does not depend on number of tokens:

$$\text{Latency} = N * FL$$

where N is the number of buffers and FL is the forward latency

In more general form,

$$\text{Latency} = \sum_{i=1}^N FL_i$$

When the pipeline is not empty, the time between arrival of the token at the input of the pipeline and appearance of the token at the output of the pipeline does depend on the number of tokens:

$$\text{Latency} = N * FL + (t - 1) * (FL + BL - prd)$$

Where t is the token and prd is the token period (spread), or one over rate of token arrival.

The latencies are shown in Fig. 1

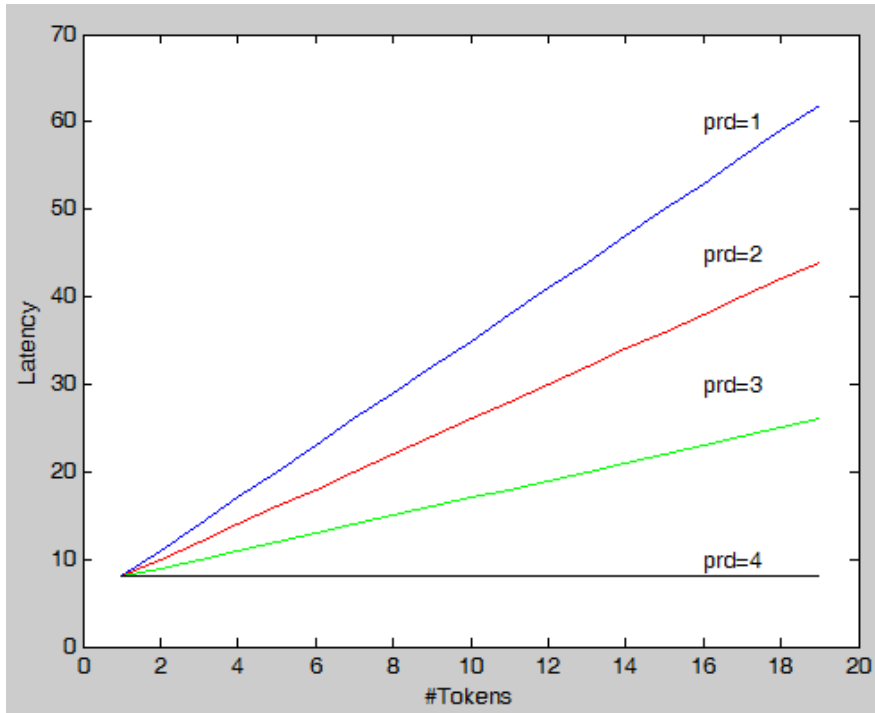


Fig. 1 Latency vs. #tokens, $N = 4, FL = \{2\ 2\ 2\ 2\}, BL = \{2\ 2\ 2\ 2\}$ for various token periods

Matlab model is attached.

Throughput vs. Frequency

I believe there's a mix-up between throughput and frequency. On page 2 of the section 4, throughput is defined as "Results (tokens) per second - inverse of cycle time". The two parts of this definition are clearly not the same. "Tokens per second" is indeed throughput while "inverse of cycle time" is in fact frequency.

Needless to say, throughput and frequency are directly proportional:

$$TP = N * F$$

Where TP is throughput, F is frequency and N is the number of buffers.

Example: If $FL = 1$, $BL = 1$ and $N = 20$, the throughput is $\frac{20}{1+1} = 10$, while if $N = 20000$ the throughput is 10000, while the frequency remains the same $\frac{1}{2}$.

I believe we should have called Y axis the frequency utilization instead of the throughput.

Throughput vs. # tokens

Throughput vs. #token can be written as follows:

$$TP = \frac{1}{FL} * n \text{ for token limited region, and}$$

$$TP = -\frac{1}{BL} * n \text{ for bubbles limited region}$$

Where n is the number of tokens

Or in generalized form

$$TP = \frac{N}{\sum FL_i} * n \text{ for token limited region, and}$$

$$TP = -\frac{N}{\sum BL_i} * n \text{ for bubbles limited region}$$

The following Fig. 2 depicts this

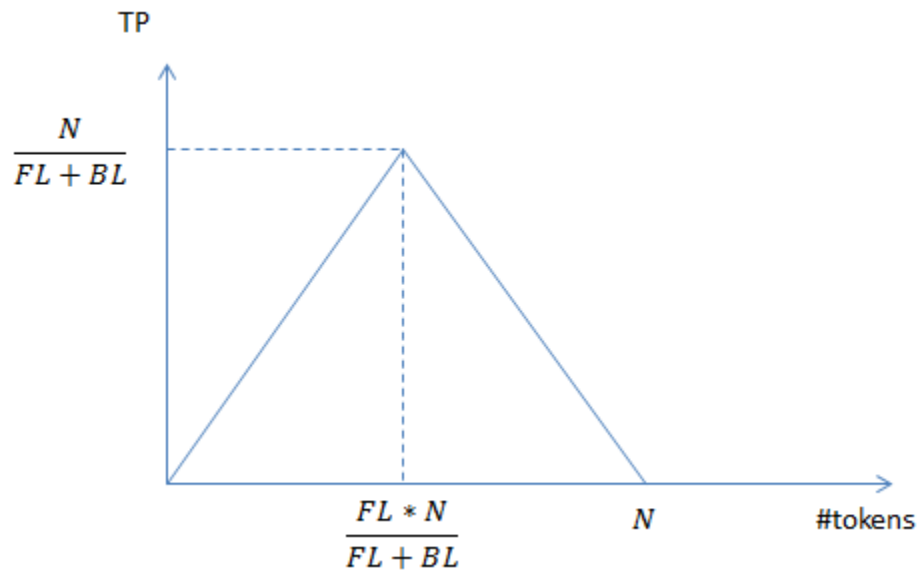


Fig. 2 Throughput vs. # tokens

When the number of buffers, increases, the throughput increases while the frequency remains constant, which can be seen in Fig. 3:

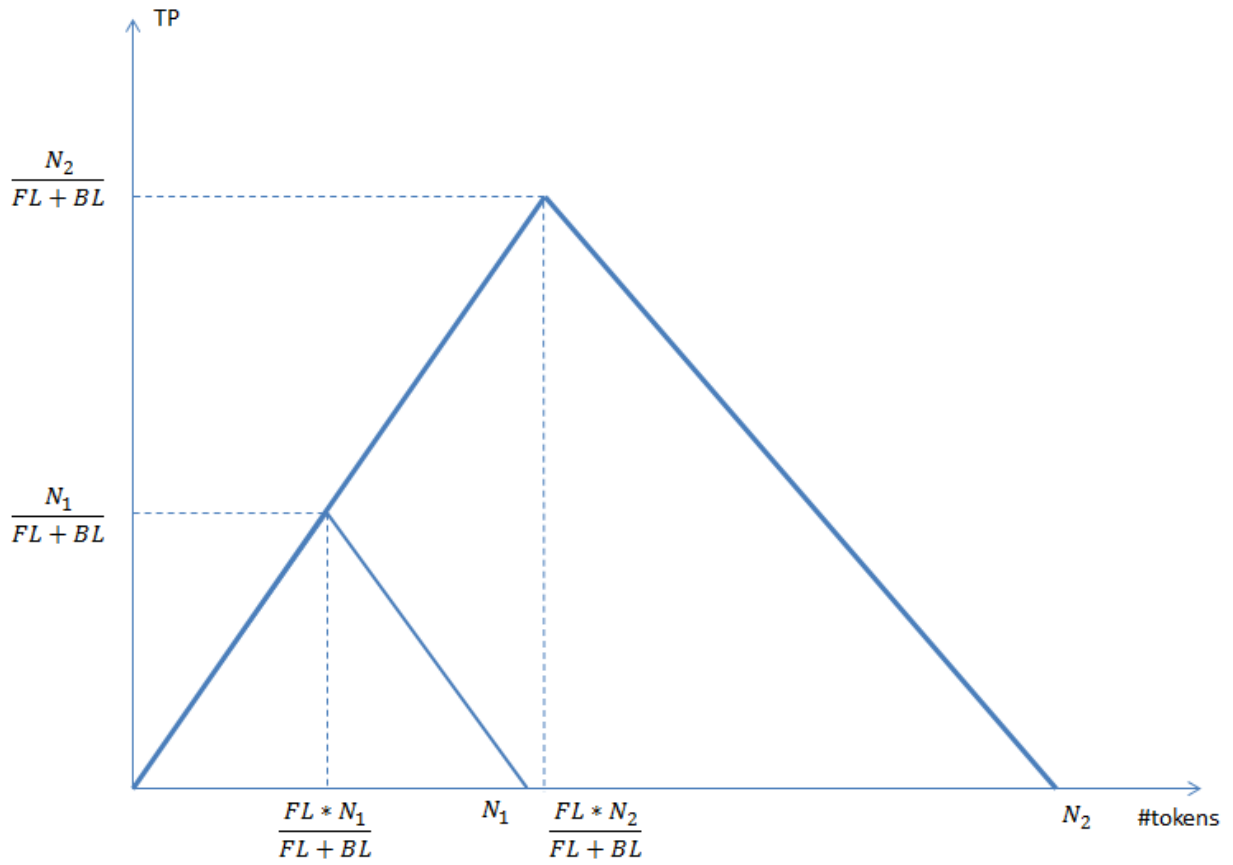


Fig. 3 Throughput vs. # tokens for various buffer sizes N

For the fork-join pipeline, the throughput of the slower branch and therefore the peak throughput of the whole pipeline can be improved by adding slack (Fig. 4):

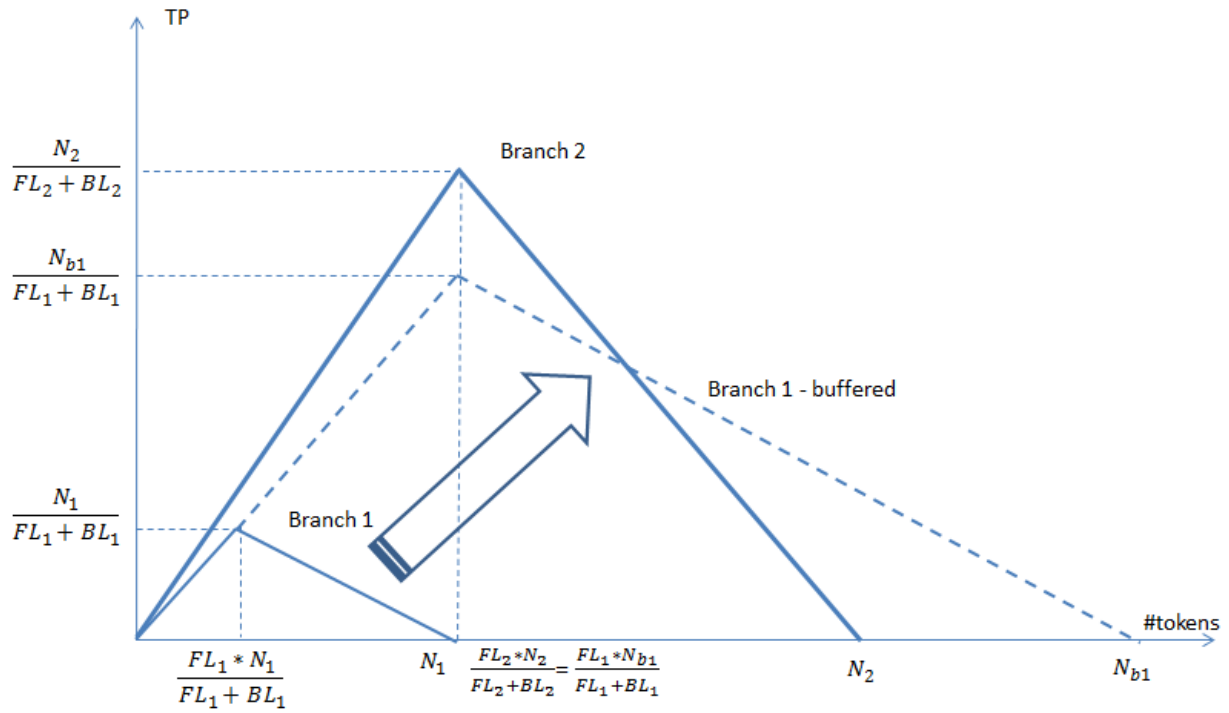


Fig. 4 Adding slack to branch 1 improves the peak performance

While adding more slack does not necessarily improves performance much further (Fig. 5):

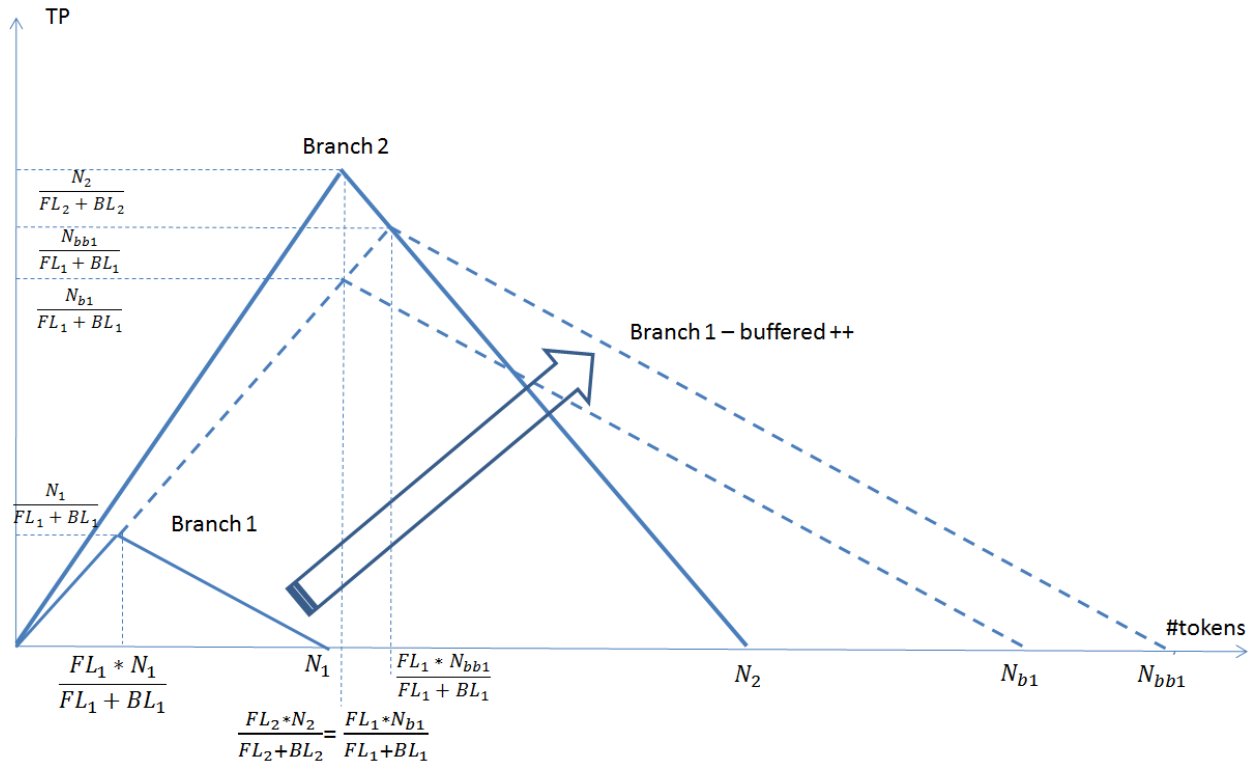


Fig. 5 Adding more slack to branch 1 does not add as much performance

```

function async_latency_calc
clear; clc
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Calculates latency of async N full-buffer pipeline
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
N=4;           % Pipe size
FL=[2 2 2 2]; % Forward latency
BL=[2 2 2 2]; % Backward latency
prd=2;        % Period of tokens (token spread)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

for i=1:N
    [latency(i,:) tokens(i,)] = async_latency(N,FL,BL,i);
end

figure(1)
plot(tokens(1,:),latency(1,:))
xlabel('#Tokens')
ylabel('Latency')
text(16,60,'prd=1')
hold
plot(tokens(2,:),latency(2,:), 'r')
text(16,45,'prd=2')
plot(tokens(3,:),latency(3,:), 'g')
text(16,30,'prd=3')
plot(tokens(4,:),latency(4,:), 'k')
text(16,10,'prd=4')
hold

function [latency tokenn]=async_latency(N,FL,BL,prd)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Calculates latency of async N full-buffer pipeline
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
print_enb=0; % Print intermediate results
runtime=20*N; % Sim time
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
hld=zeros(1,N+1);
buffer_in=zeros(1,N);
buffer_out=zeros(1,N);
token_in=zeros(1,N);
token_out=zeros(1,N);
token_generator=zeros(1,runtime);
input_token=0;
j=1;ind_out=0;ind_in=0;

for d=0:runtime
    if mod(d,prd)==0
        input_token = d+1;
        ind_in=ind_in+1;
        token_generator(ind_in)=input_token;
    end

    for k=1:N
        if hld(k)==2
            if d-token_out(k)>=BL(k) && hld(k+1)~=2
                hld(k)=0;
                token_out(k)=0;
                buffer_out(k)=0;
            end
        end
        if hld(k)==1

```

```

        if d-token_in(k)>=FL(k)
            buffer_out(k)=buffer_in(k);
            token_in(k)=0;
            token_out(k)=d;
            hld(k)=2;
        end
    end

    if hld(k)==0
        if k==1
            if ind_in>ind_out
                ind_out=ind_out+1;
                buffer_in(1)=token_generator(ind_out);
                if print_enb fprintf('[%d] >>> Token %d enters\n', d, buffer_in(1)); end
            else
                buffer_in(1)=0;
            end
        else
            buffer_in(k)=buffer_out(k-1);
            buffer_out(k-1)=0;
        end
        if buffer_in(k)>0
            token_in(k)=d;
            hld(k)=1;
        end
    end
end

end

bb=buffer_out(N);
if bb>0
    if print_enb fprintf('[%d] Token %d exits >>>>\n', d, bb); end
else
    if print_enb fprintf('[%d] No token exits \n', d); end
end

if bb>0
    latency(j)=d-(bb-1);
    tokenn(j)=(bb-1)/prd+1;
    buffer_out(N)=0;
    j=j+1;
end

end

if print_enb
    fprintf('Token      Latency\n')
    for i=1:j-1
        fprintf('%4d      %4d\n', tokenn(i), latency(i));
    end
end
end

```