1 Simulations

1.1 4x4 Crossbar

`Bestimmen und begründen Sie wie die Werte zustande kommen`

When observing the crossbar, we see certain values for minimum and maximum, as well as in between. For example the waiting time inside the arbiter queue is one of

$$t_{\text{wait}} \in \{0, 512, 1024, 1536\} = C \tag{1}$$

To get to a approximation of the avg values we then needed to weight these values by certain criteria:

- A number of Nodes is generating messages for a subset of nodes (could be from 1 to N in size) $\Rightarrow \binom{N}{n}$
- These subsets could be distributed in certain ways to the generating nodes. $\Rightarrow (N-n+1)!$

this would mean our weigth w_n is

$$w_n = \binom{N}{n}(N-n+1)! \tag{2}$$

and our avg for a number N of values C_n to avg:

$$Avg = \sum_{n=0}^{N} \frac{w_n \cdot C_n}{\sum_{k=0}^{N} w_n} \tag{3}$$

And while this may be disragarding othe reffects and randomness, it should give an apprimation within reason of the simulated values.

1.1.1 avg Input Queue Length

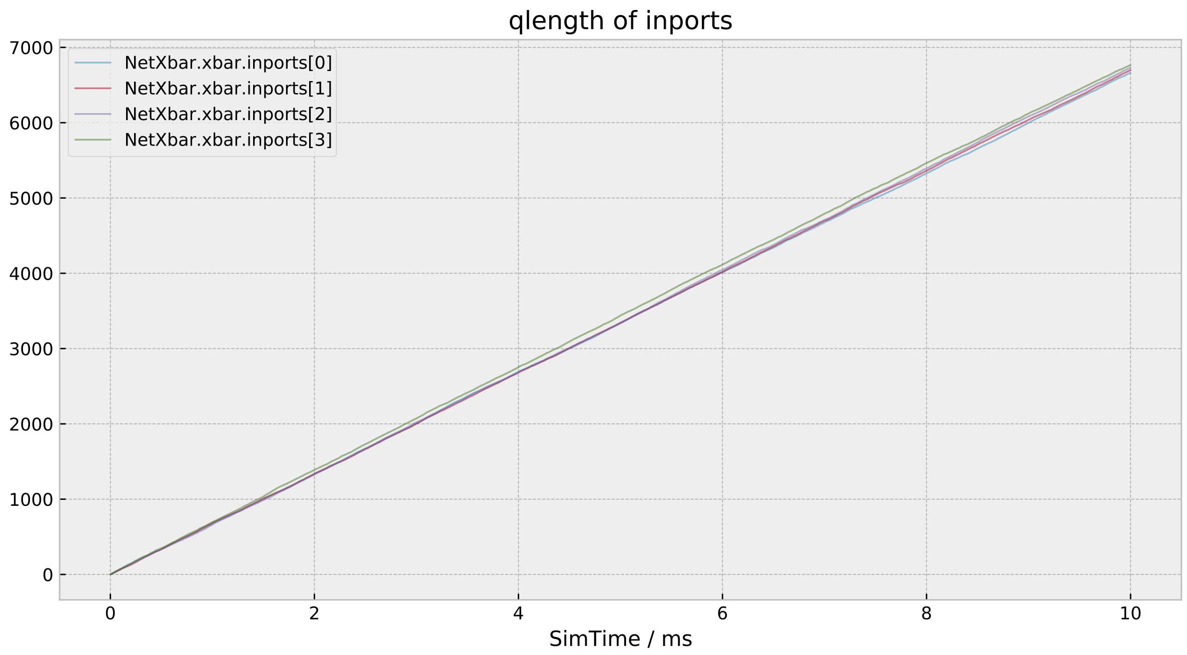

Abbildung 1: Pre Optimization

- Generated Packets
  - size: $s = 512\,\mathrm{b}$
  - send interval: $t = 512\,\mathrm{ns}$
- Connection to XBar
  - data rate: $1\,\mathrm{Gbps}$

$$r_{\text{gen}} = \frac{512\,\mathrm{b}}{512\,\mathrm{ns}} = 1\,\mathrm{Gbps} \tag{4}$$

$$C_n = 250\,\mathrm{Mbps} \cdot n \tag{5}$$

$$r_{\text{q fill,eff}} = \sum_{n=1}^{4} \frac{\binom{4}{n}(5-n)! \cdot C_n}{141} \tag{6}$$

$$= 347.5\,\mathrm{Mbps} \tag{7}$$

$$\Rightarrow l_{\text{q, avg}} = \frac{t_{\text{sim}} \cdot r_{\text{q fill, eff}}}{2 \cdot s} = 3394 \tag{8}$$

1.1.2 avg End-to-End Latency

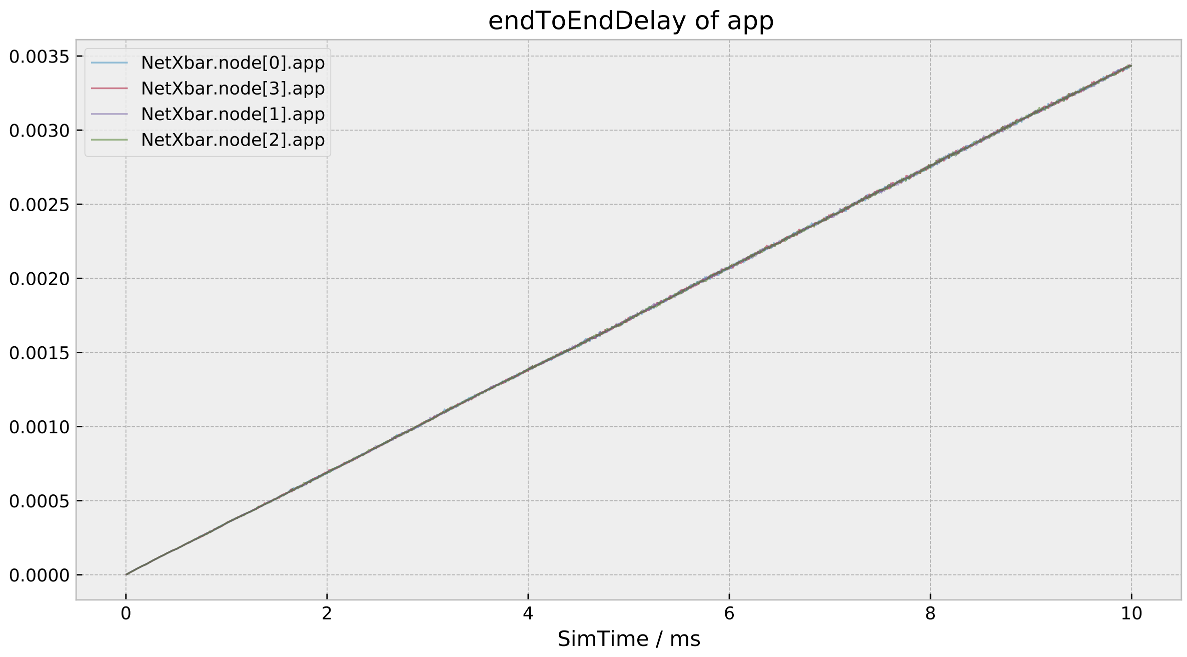

Abbildung 2: Pre Optimization e2e latency

- (no delays inside buffers, because the generated data rate, even for all 4 apps, is lower than a single data rate channels maximum throughput)
- minimum
  - App → C → Inport → C → Outport → C → App
  - delay for packet: $t_{delay} = 512ns$ (per `DatarateChannel` C)

$$\Rightarrow t_{e2e,min} = 1.536\,\mu\text{s} \tag{9}$$

- maximum

  at end of simulation, inport buffer full, all in to one out

$$l_{inportq,max} = 7031 \tag{10}$$
$$r_{dequeue,min} = 250\,\text{Mbps} \tag{11}$$
$$t_{\text{in queue, max}} = \frac{t_{q,inport} \cdot s}{r_{dequeue}} = 3.599\,\text{ms} \tag{12}$$
$$\tag{13}$$

- on avg

$$t_{\text{e2e, avg}} = t_{\text{in queue, max}}/2 = 1.8\,\text{ms} \tag{14}$$

1.1.3 avg Arbiter Request Queue Length

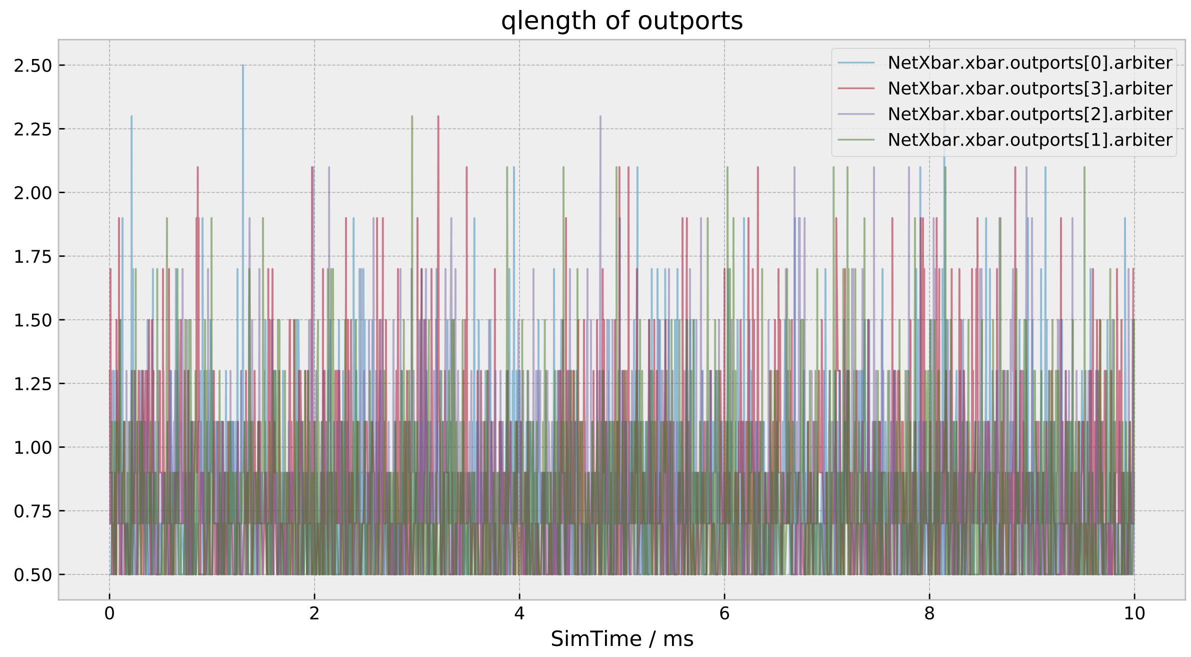

Abbildung 3: Pre Optimization Queue Length

- cases

$$C_n = n - 1 \qquad 0 \leq n < 4 = N \tag{15}$$

  where C_n is the number of waiting packets

- on avg (analogously to t_{e2e})

$$\Rightarrow l_{arbq,avg} = \sum_{n=1}^{4} \frac{\binom{4}{n}(5-n)! \cdot C_n}{141} = 0.39 \tag{16}$$

1.1.4 avg Arbiter Request Queue Time

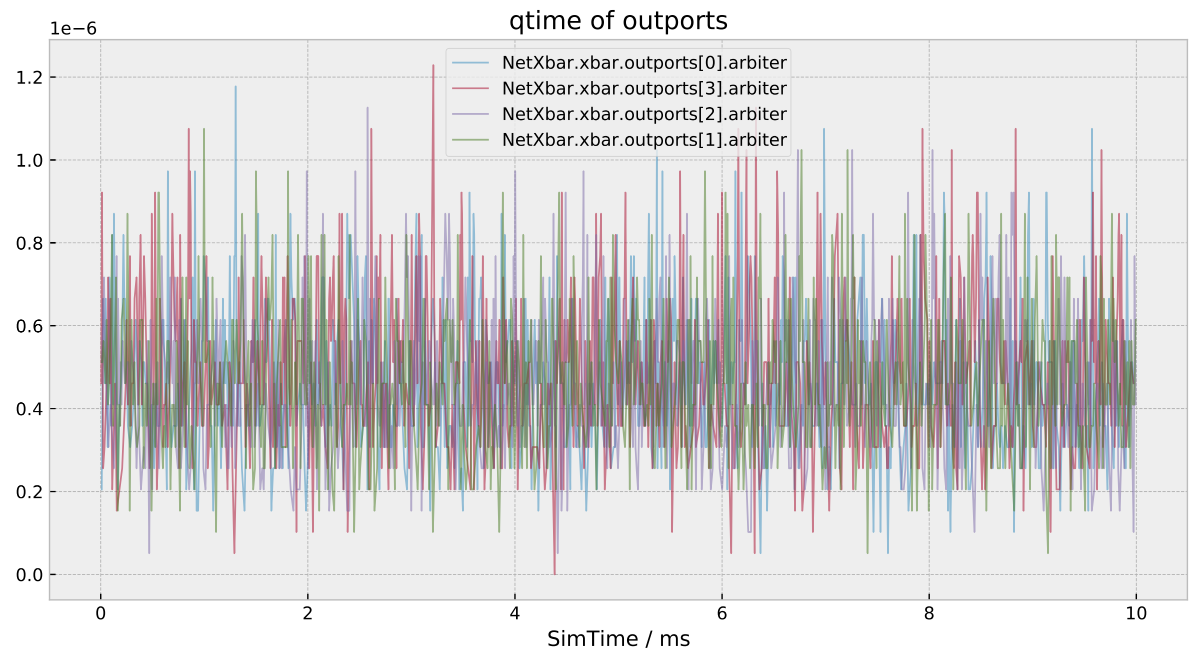

Abbildung 4: Pre Optimization Arbiter Queue Times

- cases

$$C_n = 512\,\mathrm{ns} \cdot n \tag{17}$$

- on avg

$$\Rightarrow t_{arbq,avg} = \sum_{n=1}^{4} \frac{\binom{4}{n}\,(5-n)! \cdot C_n}{141} \tag{18}$$

$$= 711\,\mathrm{ns} \tag{19}$$

1.1.5 avg Output Buffer Queue Length

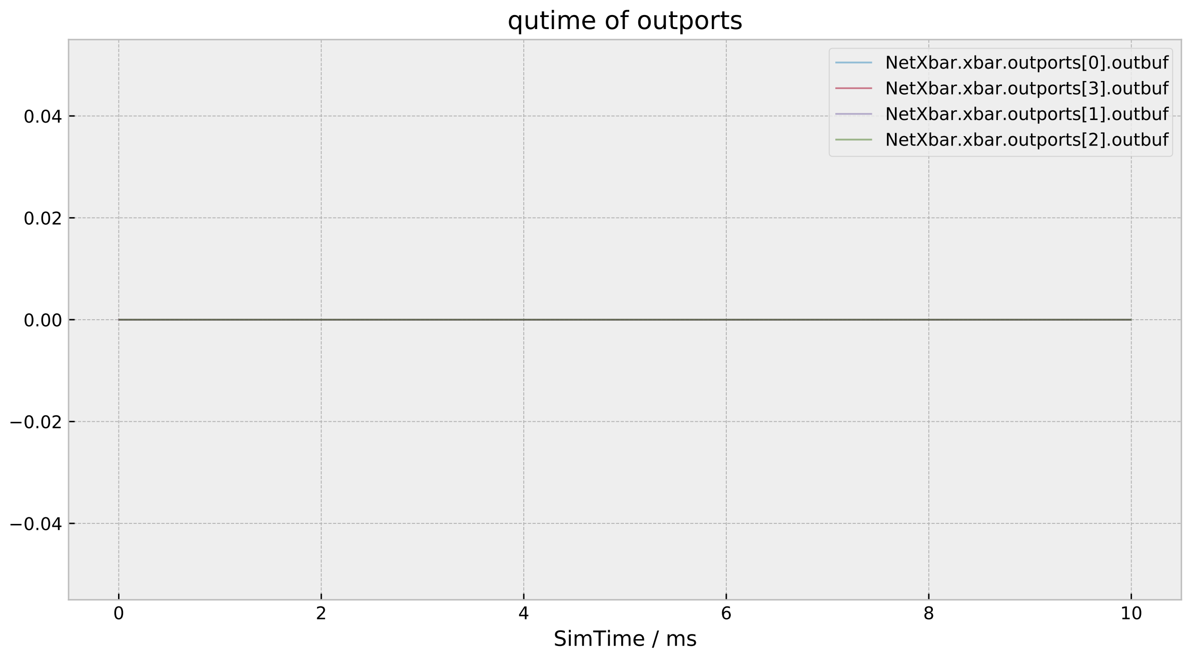

Abbildung 5: Pre Optimization Output buffer Queue Time

- Generated Packets
  - size: $s = 512\,\mathrm{b}$
  - send interval: $t = \mathrm{uniform}(1\,\mu\mathrm{s}, 10\,\mu\mathrm{s}) = 5.5\,\mu\mathrm{s}$
- Connection to and in XBar
  - data rate: $1\,\mathrm{Gbps}$

$$r_{\text{generated}} = \frac{512\,\mathrm{b}}{5.5\,\mu\mathrm{s}} = 93\,\mathrm{Mbps} \tag{20}$$
$$\Rightarrow l_{\text{q,avg}} = 0 \tag{21}$$

1.1.6 avg Throughput

This calculation is based on the inport queue fillrate that we estimated using our formula

$$r_{\text{tp, eff}} = 4 \cdot (r_{\text{gen}} - r_{\text{q fill, eff}}) \tag{22}$$
$$= 4 \cdot (1000\,\mathrm{Mbps} - 372\,\mathrm{Mbps}) \tag{23}$$
$$= 2512\,\mathrm{Mbps} \tag{24}$$

1.2 Throughput vs. Ports

Here we caclulated the theoretical throughput and compared it with the simulated, based on the inport fillrate.

# Nodes	$r_{\text{tp, eff}}$	$r_{\text{tp, sim}}$
2	700 Mbps	752 Mbps
4	653 Mbps	655 Mbps
8	627 Mbps	616 Mbps
16	616 Mbps	603 Mbps
32	610 Mbps	591 Mbps

1.3 Throughput vs. Injection Rate

- not in saturation until delay $\leq 832\,\text{ns}$
- saturation point $r_{sat} = \frac{512\,\text{b}}{832\,\text{ns}} = 615\,\text{Mbps}$
- The main reason will be the arbiter. It can only process packets at about 600 Mbps on avg., therefore bottlenecking the rest of the system

1.4 Throughput vs. Bandwidth

- Network in saturation until bandwidth $> 1600\,\text{Mbps}$
- throughput at saturation point $r_{sat} = 1\,\text{Gbps}$
- If the Arbiter can only work at $\approx 62\%$ of the bandwidth, we reach this saturation point if 62% of the bandwidth is 1 Gbps, therefore the required bandwith is 1613 Gbps

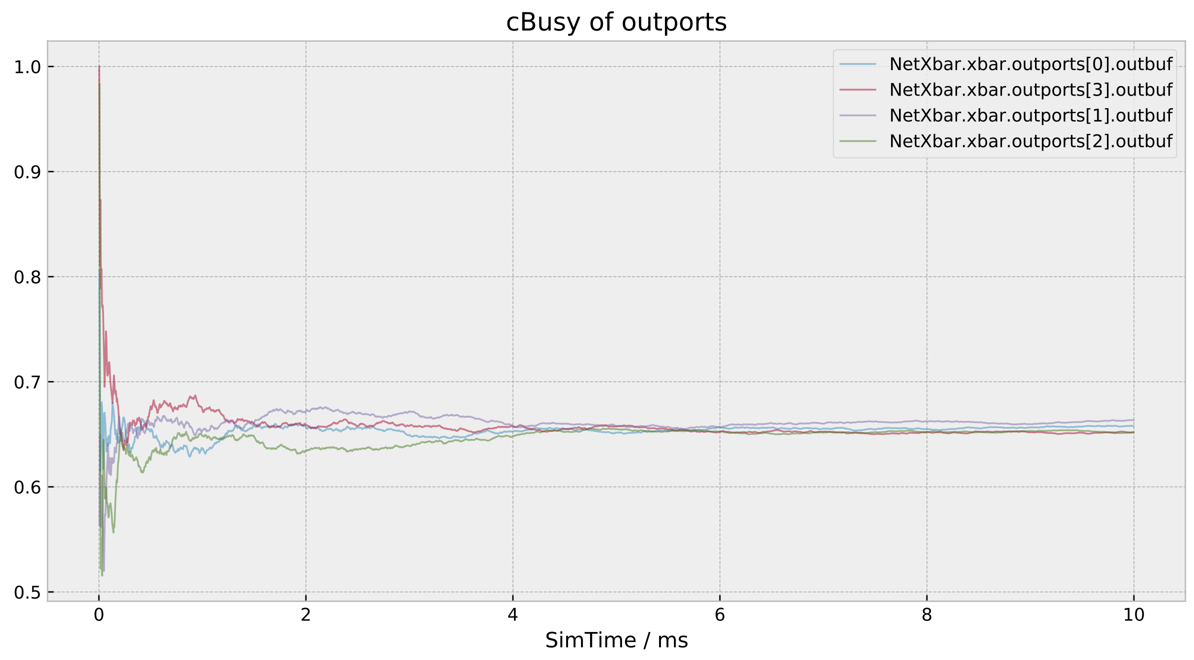

Abbildung 6: Busy state of Output Buffer out

2 Optimizations

2.1 Problems

In the previous exercise we determined the arbiter to be the limiting resource. We determined head of line blocking to be a limiting factor for performance. Since packets are sent to random nodes it is very likely for a packet having to wait because the route is currently occupied even though other packets in queue could be routed. Therefore virtual queues were implemented.

2.2 Solution

We can measure the performance gain in correlation to the average inport buffer length which is decreased by two orders of magnitude. Form around 3000 to around 15

2.3 Problems In Real Hardware

In a real hardware setting this would increase complexity. Either distinct buffers have to be used or some sort of control logic that holds pointers to the virtual queues. Spatial constraints have to be taken into account as well as scalability. With more possible routing decisions the number of virtual queues also rises.

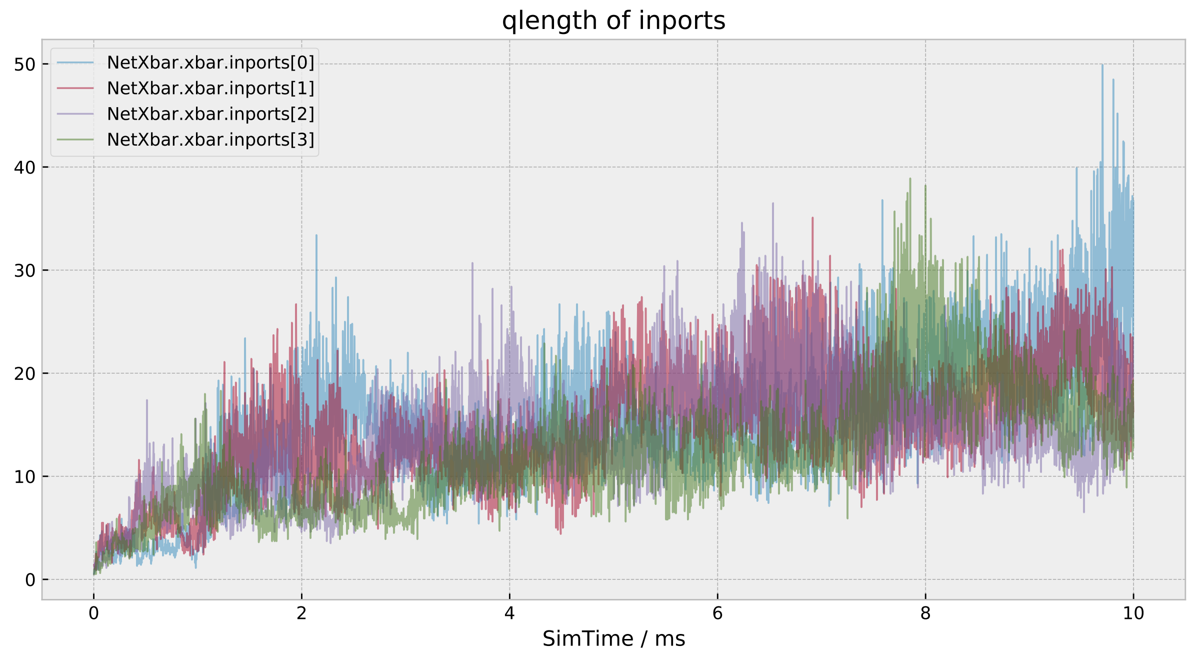

Abbildung 7: Post optimization Inport Queue Length

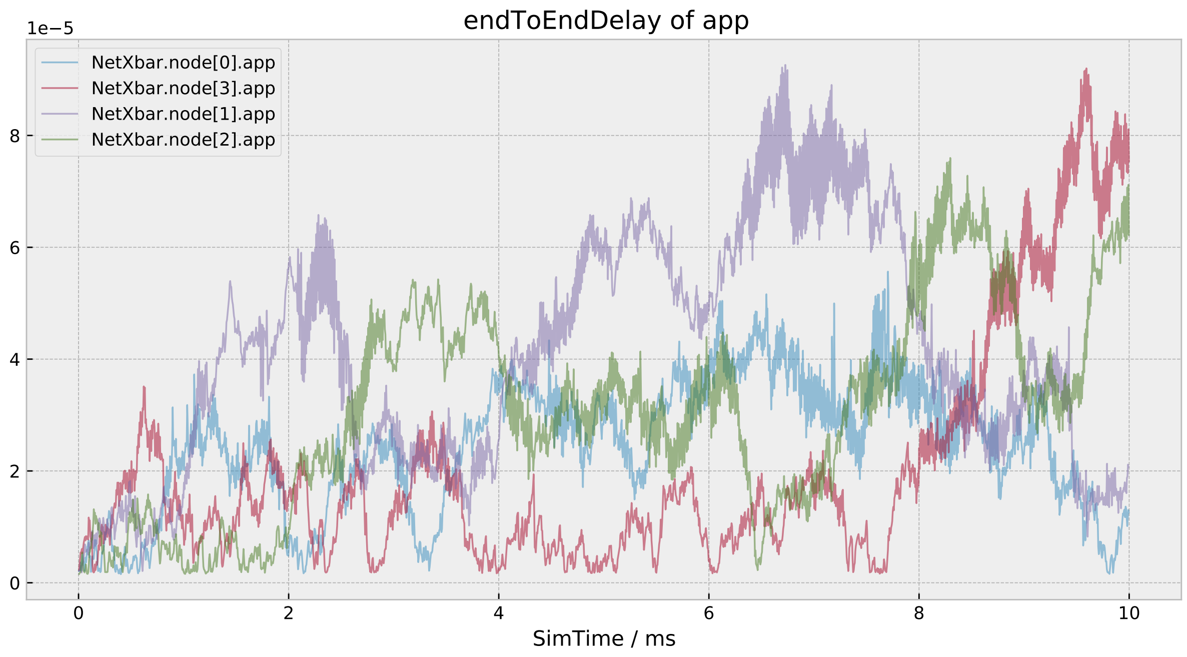

Abbildung 8: Post Optimization App E2E Delay

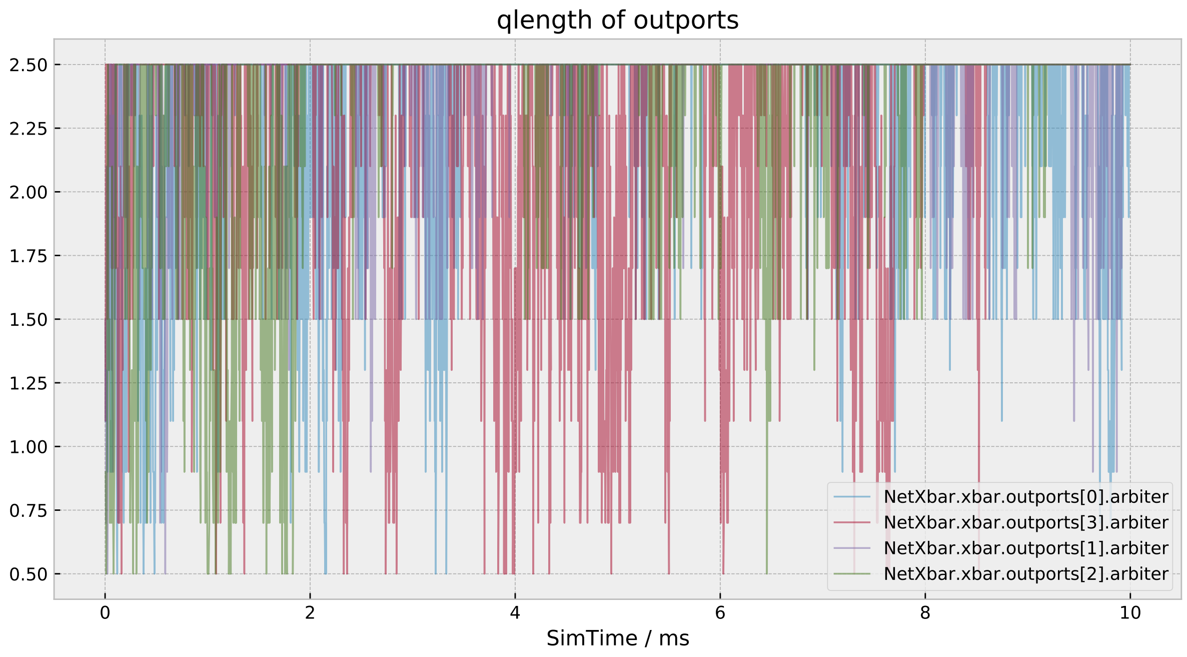

Abbildung 9: Post Optimization Arbiter Queue Length

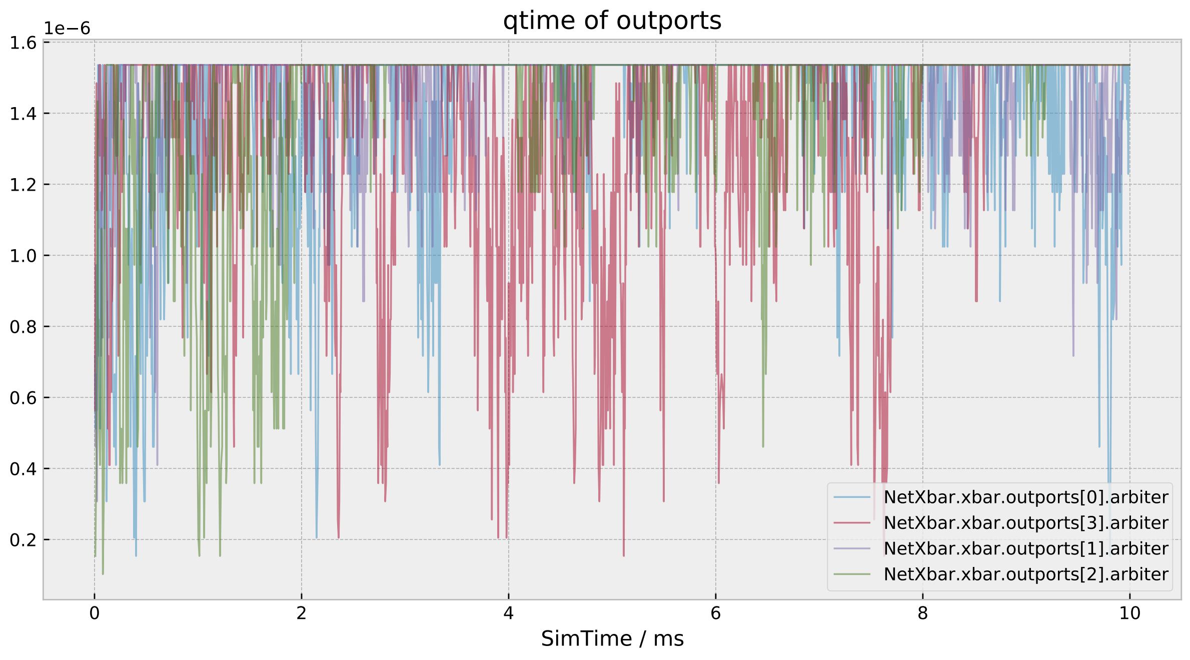

Abbildung 10: Post Optimization Arbiter Queue Time

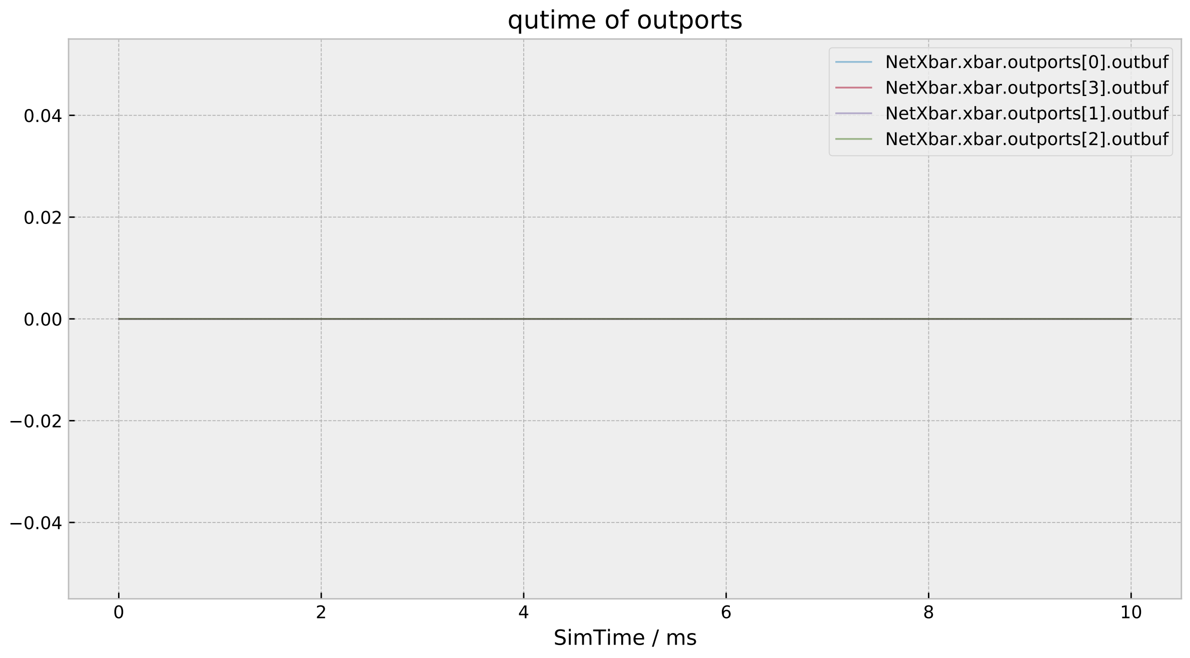

Abbildung 11: Post Optimization Output Queue Time

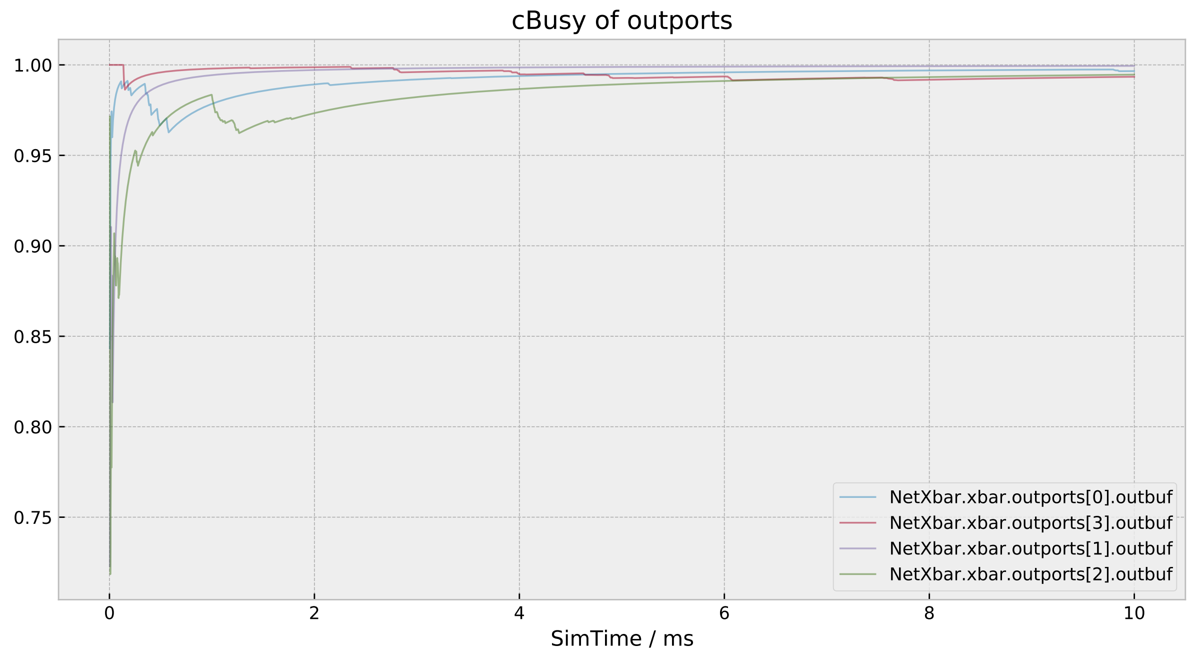

Abbildung 12: Post optimization busy time