1 Simulations

1.1 4x4 Crossbar

`Bestimmen und begründen Sie wie die Werte zustande kommen`

$$Avg = \sum_{n=0}^{N} \frac{\binom{N}{n}(N-n+1)! \cdot C_n}{\sum_{k=0}^{N}\binom{N}{k}(N-k+1)!} \tag{1}$$

- `avg Input Queue Length`
  - Generated Packets
    - size: $s = 512\,\text{b}$
    - send interval: $t = 512\,\text{ns}$
  - Connection to XBar
    - data rate: $1\,\text{Gbps}$

$$r_{\text{gen}} = \frac{512\,\text{b}}{512\,\text{ns}} = 1\,\text{Gbps} \tag{2}$$

$$C_n = 250\,\text{Mbps} \cdot n \tag{3}$$

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

$$= 347.5\,\text{Mbps} \tag{5}$$

$$\Rightarrow l_{\text{q,avg}} = \frac{t_{\text{sim}} \cdot r_{\text{q fill, eff}}}{2 \cdot s} = 3394 \tag{6}$$

- `avg End-to-End 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)

$$\Rightarrow t_{e2e,min} = 1.536\,\mu\text{s} \tag{7}$$

  - maximum

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

$$l_{inportq,max} = 7031 \tag{8}$$

$$r_{dequeue,min} = 250\,\text{Mbps} \tag{9}$$

$$t_{\text{in queue, max}} = \frac{t_{q,inport} \cdot s}{r_{dequeue}} = 3.599\,\text{ms} \tag{10}$$

$$\tag{11}$$

  - on avg

$$t_{\text{e2e, avg}} = t_{\text{in queue, max}}/2 = 1.8\,\text{ms} \tag{12}$$

- `avg Arbiter Request Queue Length`
  - cases

$$C_n = n - 1 \qquad 1 \leq n \leq 4 = N \tag{13}$$

  - on avg (analogously to t_{e2e})

$$\Rightarrow l_{arbq,avg} = \sum_{n=1}^{4} \frac{\binom{4}{n}\,(5-n)! \cdot C_n}{141} = 0.39 \tag{14}$$

- `avg Arbiter Request Queue Time`
  - cases

$$C_n = 512\,\text{ns} * n \tag{15}$$

  - on avg

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

$$= \tag{17}$$

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

$$r_{\text{generated}} = \frac{512\,\text{b}}{5.5\,\mu\text{s}} = 93\,\text{Mbps} \tag{18}$$

$$\Rightarrow l_{\text{q,avg}} = 0 \tag{19}$$

- `avg Throughput`

  input and output buffers always empty

$$r_{cross} = 4 * r_{\text{gen}} \tag{20}$$

$$= 372\,\text{Mbps} \tag{21}$$

1.2 Throughput vs. Ports

- with our formula 0.6, 0.44, 0.347518, 0.286498, 0.243324, 0.211277, 0.186604, 0.167047, 0.151176, 0.138045,

  0.127005, 0.117594, 0.109478, 0.102407, 0.0961928, 0.0906881, 0.0857783, 0.0813722, 0.077396,

  0.0737899, 0.0705046, 0.067499, 0.0647391, 0.0621958, 0.0598446, 0.0576646, 0.0556378, 0.0537485,

  0.0519832, 0.0503302, 0.048779

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

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 1: sim

Abbildung 2

Abbildung 3