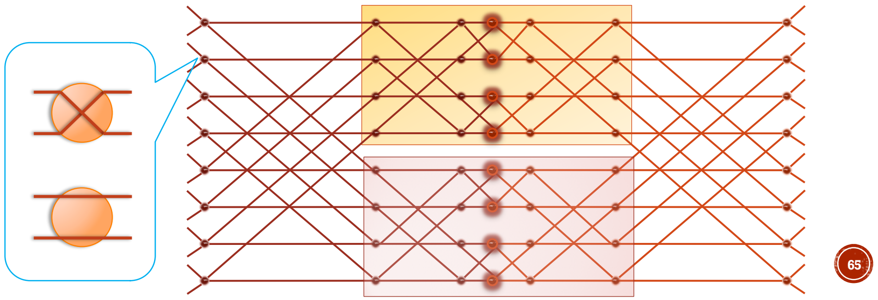
BENEŠ NETWORK (8)

PROOF OF THE REARRANGEABILITY OF THE BENEŠ NETWORK (CNTD)

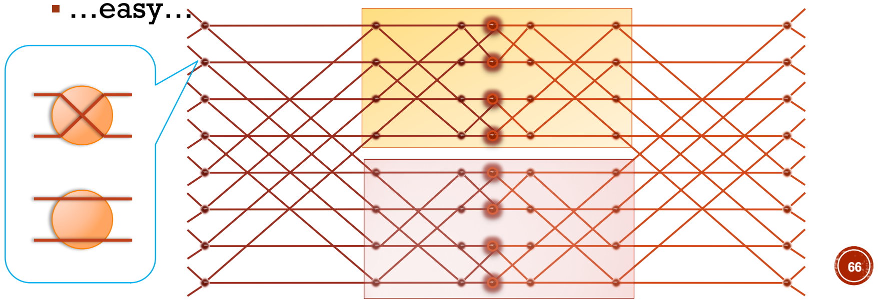
- Hence, for each path, it will be sufficient to decide whether it is to be routed through the upper sub-Beneš network or through the lower sub-Beneš network.



BENEŠ NETWORK (9)

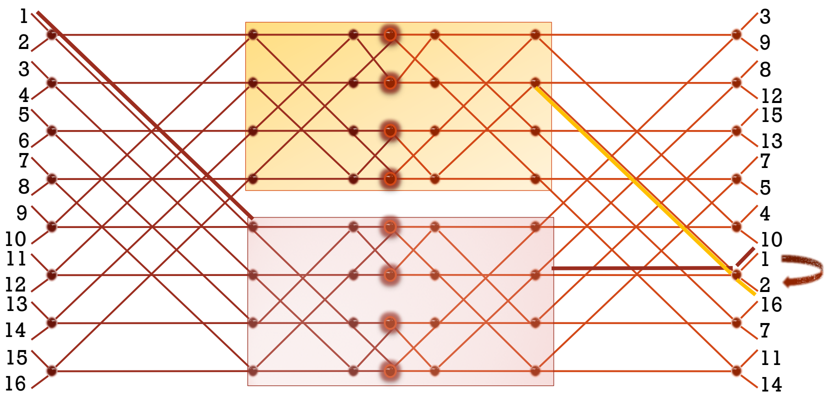
PROOF OF THE REARRANGEABILITY OF THE BENEŠ NETWORK (CNTD)

- The only constraints we have to consider to decide whether paths use the upper or lower subnetworks are that paths from inputs *2i-1* and *2i* must use different subnetworks for $1 \leq i \leq 2n$, and that paths to outputs *2i-1* and *2i* must use different subnetworks.
- …easy…



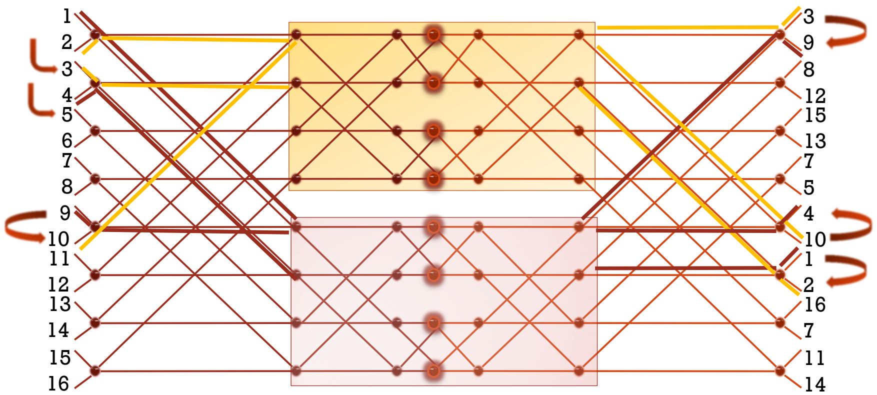
BENEŠ NETWORK (10)

PROOF OF THE REARRANGEABILITY OF THE BENEŠ NETWORK (CNTD)



BENEŠ NETWORK (11)

PROOF OF THE REARRANGEABILITY OF THE BENEŠ NETWORK (CNTD)



And so on…

BENEŠ NETWORK (12)

PROOF OF THE REARRANGEABILITY OF THE BENEŠ NETWORK (CNTD)

Summary of the steps:

- We start by routing the first path through the upper sub-network.
- We next satisfy the constraint generated at the output by routing the corresponding path through the lower sub-network.
- We keep on going back and forth through the network, satisfying constraints at the inputs by routing through the upper sub-network and satisfying constraints at the outputs by routing through the lower sub-network.
- …

BENEŠ NETWORK (13)

PROOF OF THE REARRANGEABILITY OF THE BENEŠ NETWORK (CNTD)

- …
- Eventually, we will close the loop by routing a path through the lower sub-network (in response to an output constraint) that shares an input switch with the first path that was routed.
- If any additional paths needs to be routed, we con- tinue as before, starting over again with an arbitrary unrouted path.
- In this way, all paths can be assigned to the upper or lower sub-networks without conflict.

BENEŠ NETWORK (14)

PROOF OF THE REARRANGEABILITY OF THE BENEŠ NETWORK (CNTD)

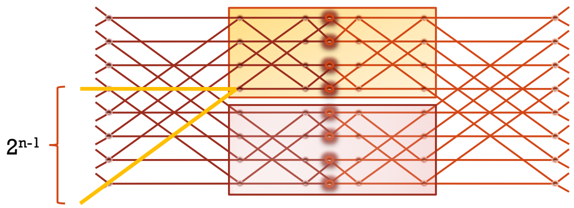
- This algorithm is called looping algorithm.
- It is easy to see that all paths can be assigned to the upper or lower sub-networks without conflict:
- By construction, if we start going to the upper sub-network, we will arrive to the corresponding output in the upper sub-network and we will leave it to the lower sub-network, and so on.
- For parity reason, when a loop is close, we will correctly arrive from the right sub-network.
- The remainder of the path routing and switch setting is handled by induction in the sub-networks.

BENEŠ NETWORK (15)

- In the case that each layer 0 node of the n-dimensional Beneš network has just one input and each layer $2n$ node has just one output, then the paths from the inputs to the outputs can be constructed so as to be node-disjoint (instead of only edge-disjoint):
- …

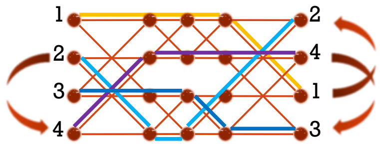
BENEŠ NETWORK (16)

- Th. *Given any one-to-one mapping of π of 2^n inputs to 2^n outputs in an n-dim. Beneš network, there is a set of node-disjoint paths from the inputs to the outputs connecting input i to output $\pi(i)$ for $1 \le i \le 2^n$.*
- Proof. Identical to the previous one, but the paths needing to use different Beneš networks are now i and $i+2^{n-1}$, $1 \le i \le 2^{n-1}$ (and not $2i-1$ and $2i$). ■



BENEŠ NETWORK (17)

- Exemple: $n=2$, hence $2^{n-1}=2$



BENEŠ NETWORK (18)

- The only drawback to these theorems is that we do not know how to set the switches on-line. In other words, each switch needs to be told what to do by a **global control** that has knowledge of the permutation being routed.
- There exist numerous methods for overcoming this difficulty (not studied here).