Port Constraints in Hierarchical Layout of Data Flow Diagrams

Miro Spönemann[1]

Hauke Fuhrmann[1] Reinhard von Hanxleden[1] Petra Mutzel[2]

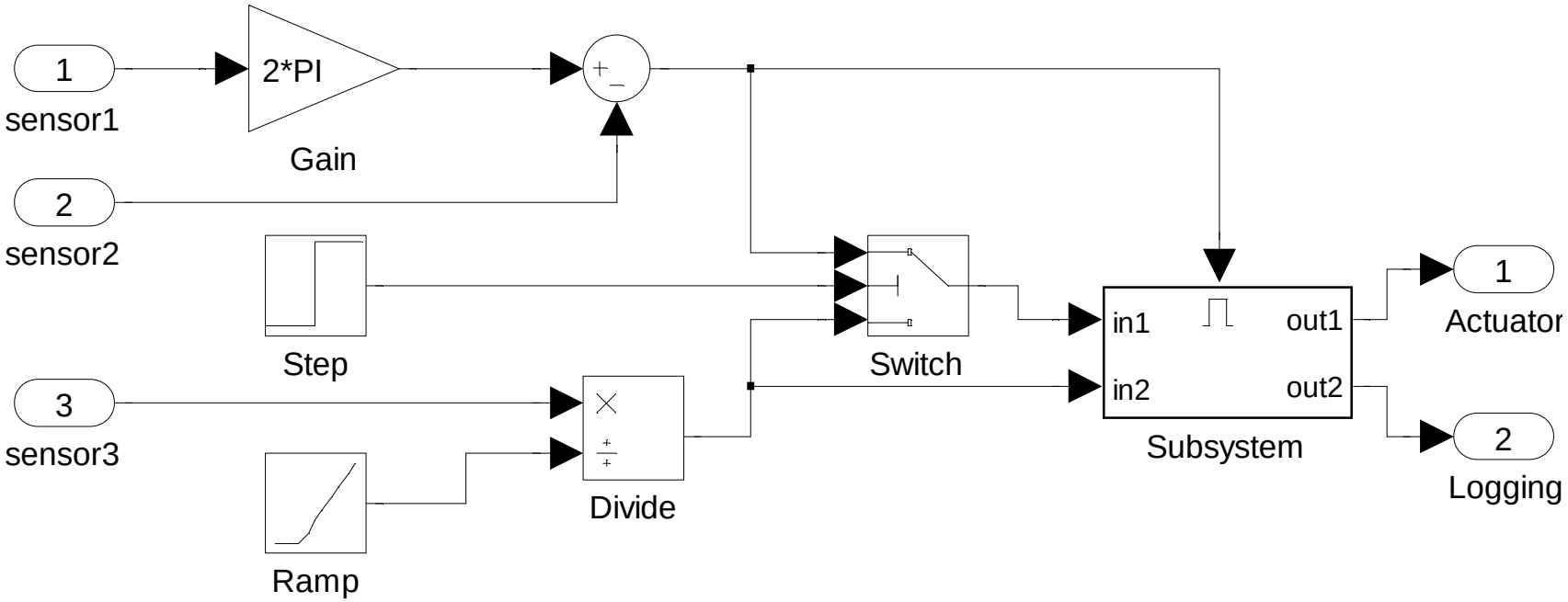
[1]Real-Time and Embedded Systems Group, Christian-Albrechts-Universität zu Kiel
{msp,haf,rvh}@informatik.uni-kiel.de

[2]Chair of Algorithm Engineering, Technische Universität Dortmund
petra.mutzel@tu-dortmund.de

Graph Drawing 2009
September 24th

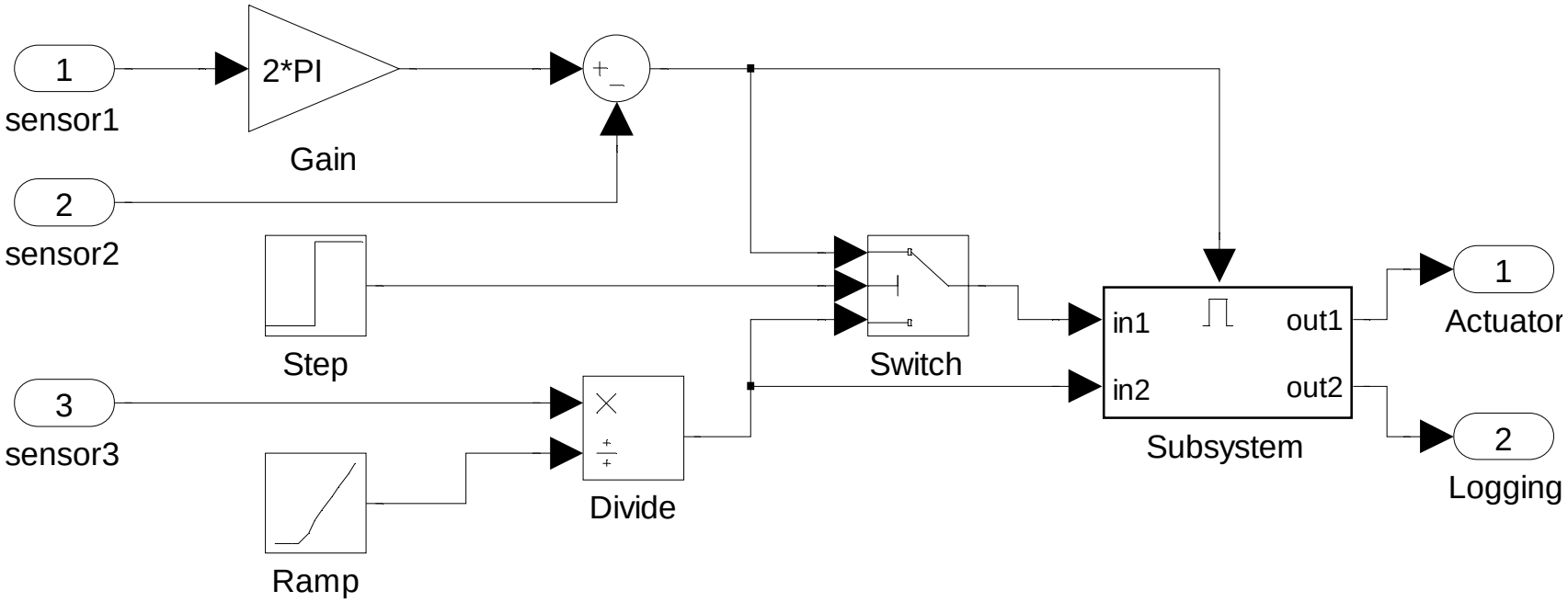
Outline

Data Flow Diagrams



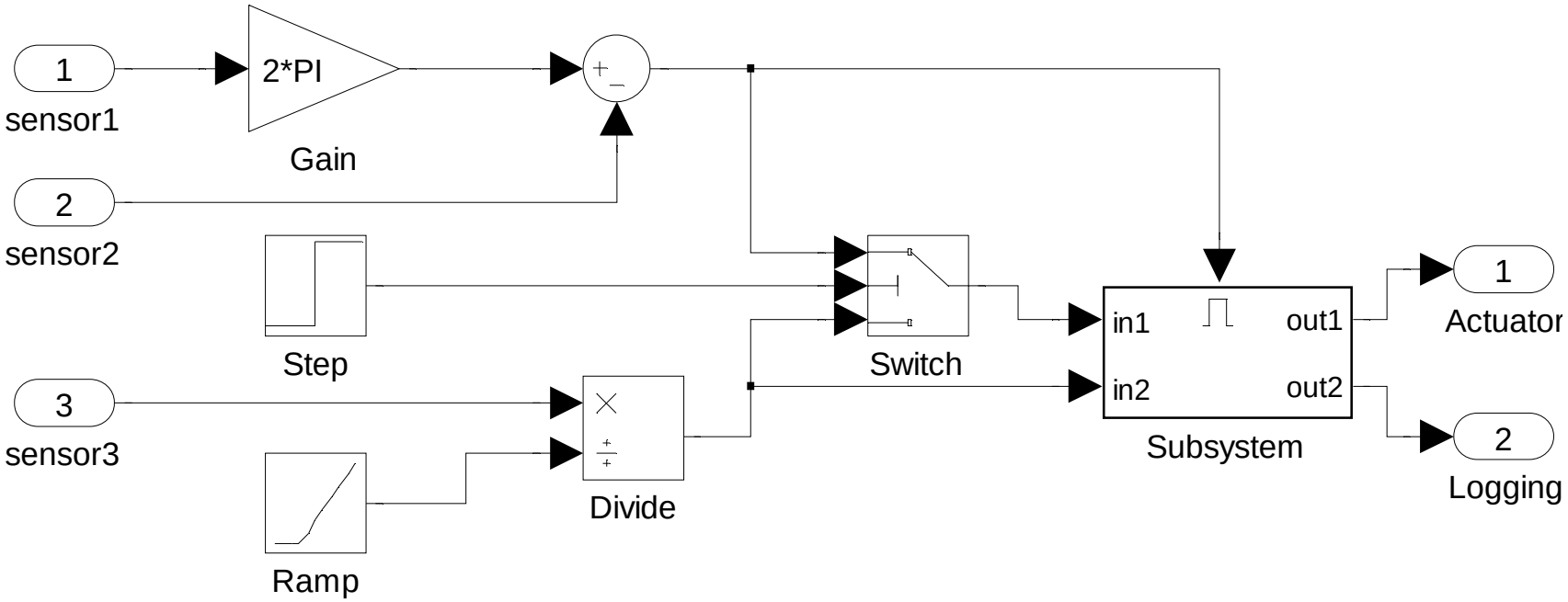
- Graphical modeling languages for the design of complex (embedded) systems

Data Flow Diagrams



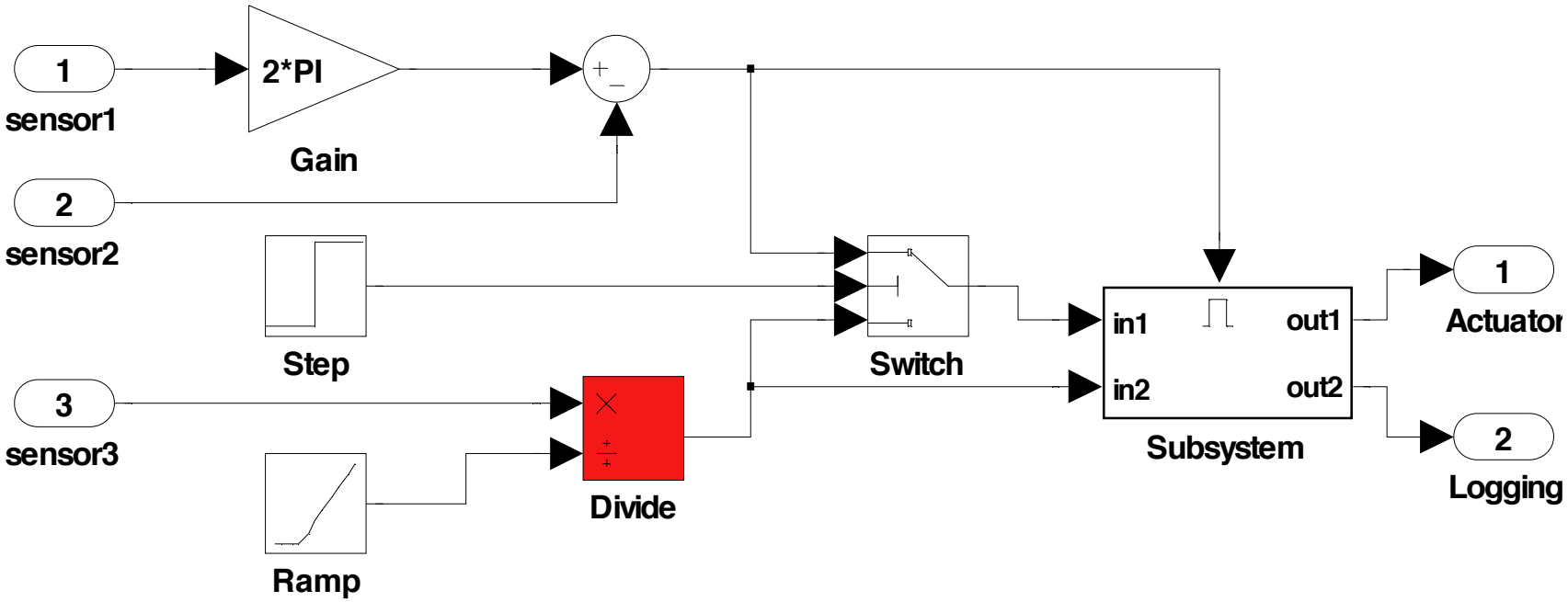
- Graphical modeling languages for the design of complex (embedded) systems
- Employed in software and hardware development

Data Flow Diagrams



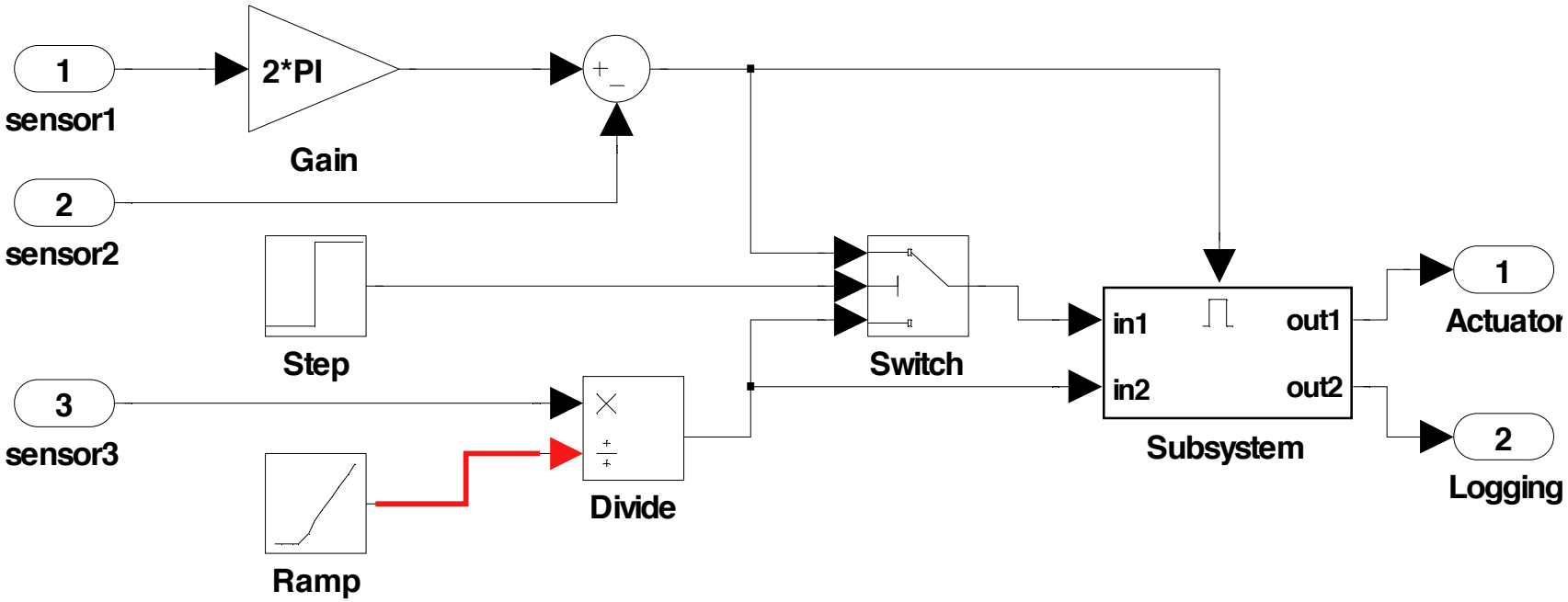
- Graphical modeling languages for the design of complex (embedded) systems
- Employed in software and hardware development
- Examples: Simulink, LabVIEW, ASCET, SCADE, Ptolemy

Data Flow Diagrams: Components



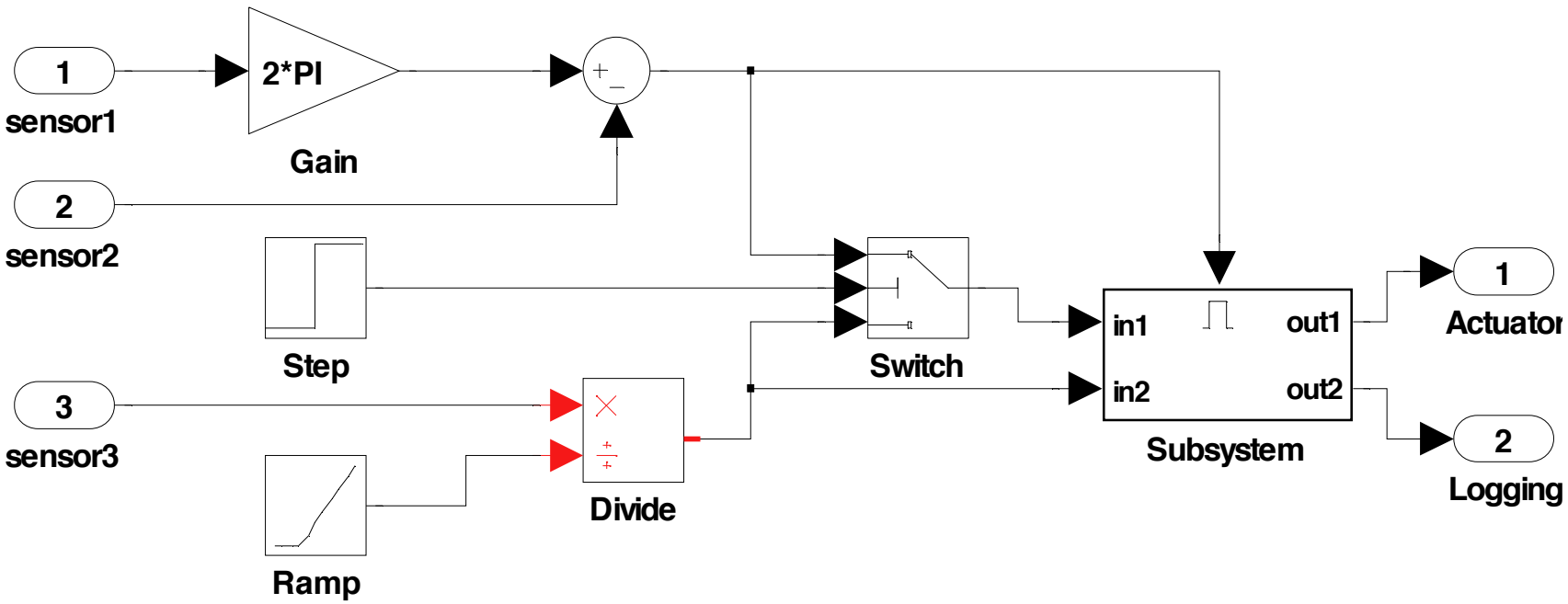
- Vertices: operators that produce and consume data

Data Flow Diagrams: Components



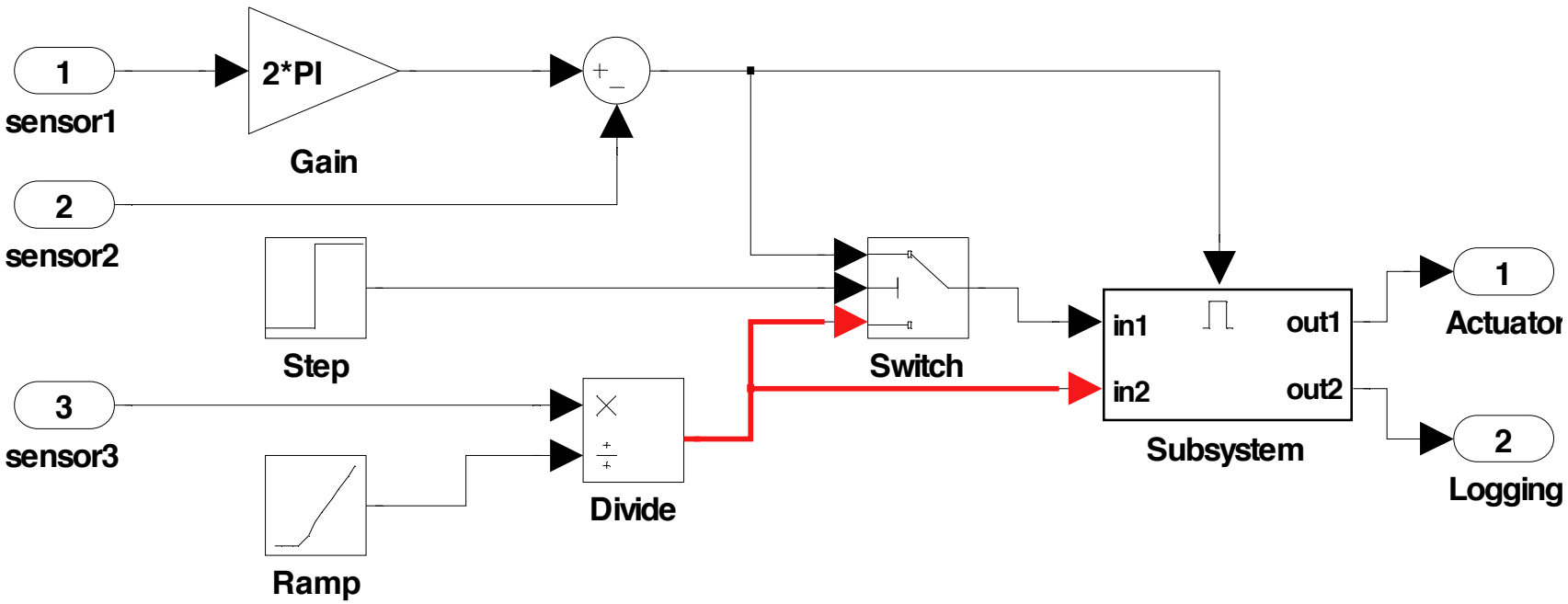
- Vertices: operators that produce and consume data
- Edges: data paths between operators (orthogonal, left to right)

Data Flow Diagrams: Components



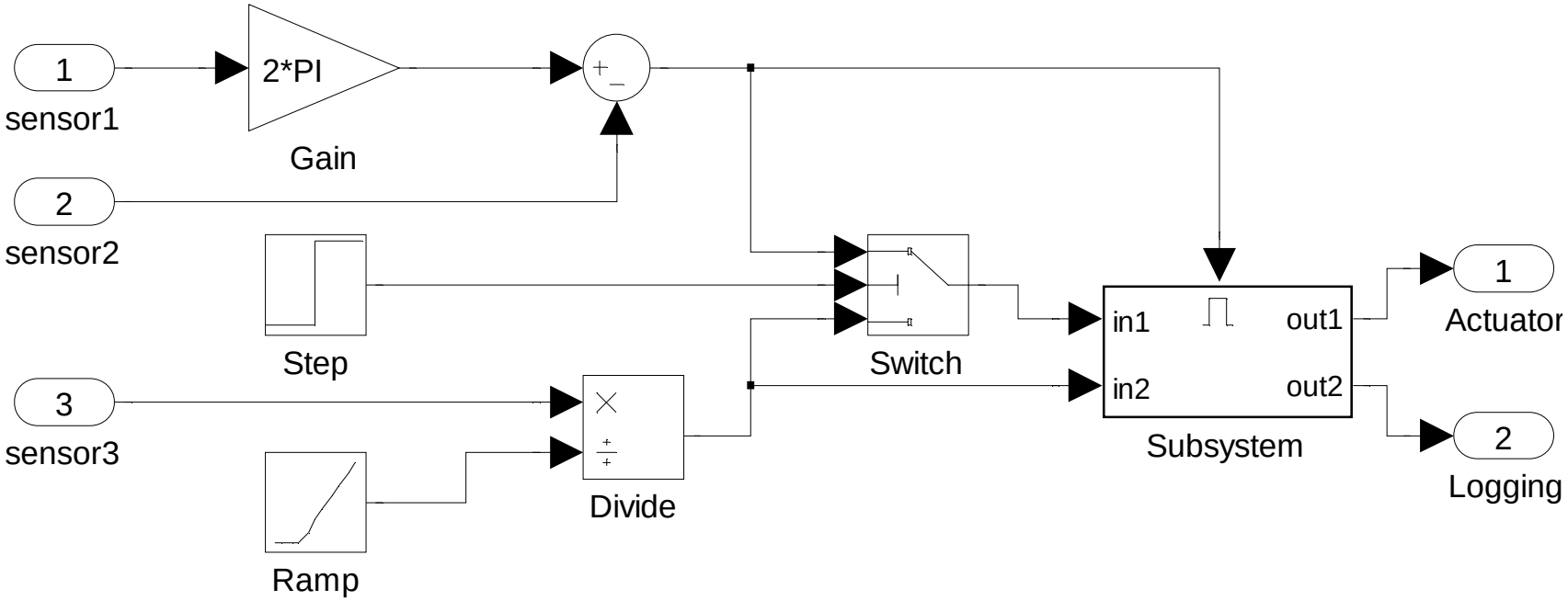
- Vertices: operators that produce and consume data
- Edges: data paths between operators (orthogonal, left to right)
- Ports: interface of the operators
  - Input ports mostly on left side, output ports on right side

Data Flow Diagrams: Components



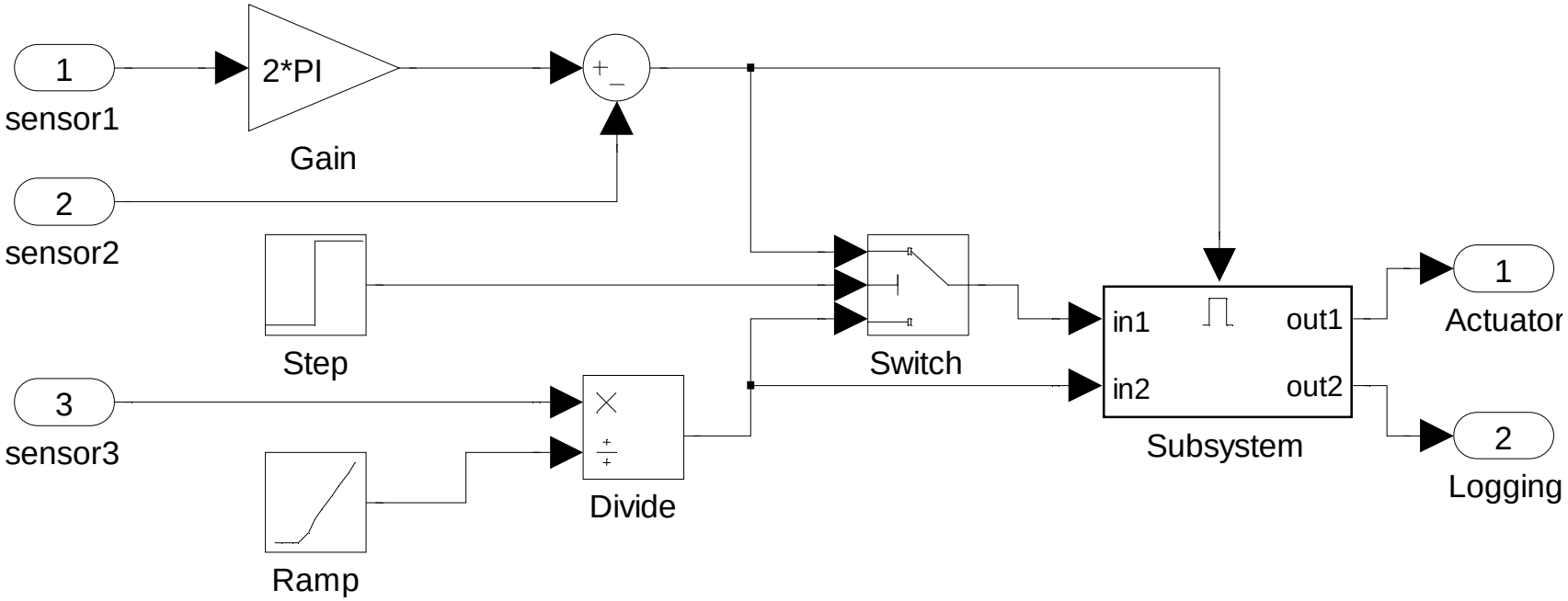
- Vertices: operators that produce and consume data
- Edges: data paths between operators (orthogonal, left to right)
- Ports: interface of the operators
  - Input ports mostly on left side, output ports on right side
- Multiple target ports lead to hyperedges

Data Flow Diagrams: Automatic Layout



- Most data flow modeling tools do not offer automatic layout

Data Flow Diagrams: Automatic Layout



- Most data flow modeling tools do not offer automatic layout
- Standard layout algorithms cannot be applied
  - Need to extend for handling of ports and hyperedges

Port Constraints

Four scenarios for constraints on port positions:

Port Constraints

Four scenarios for constraints on port positions:

1. Free ports: arbitrary positions on the node's border

Port Constraints

Four scenarios for constraints on port positions:

1. Free ports: arbitrary positions on the node's border
2. Fixed sides: top, bottom, left, or right side

Port Constraints

Four scenarios for constraints on port positions:

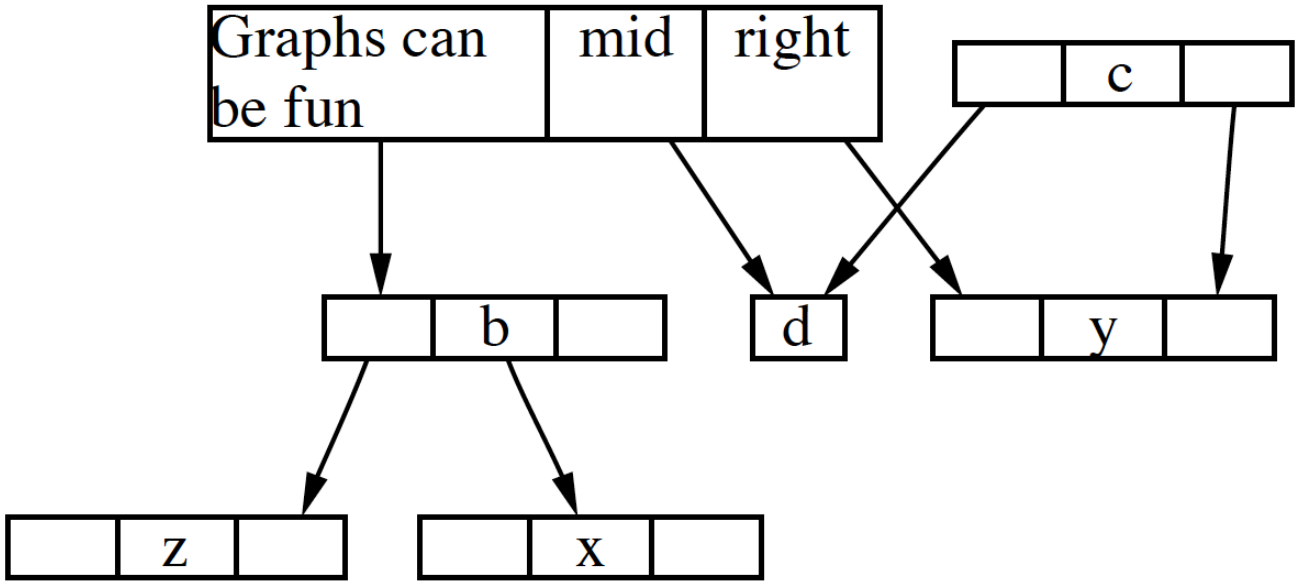
1. Free ports: arbitrary positions on the node's border
2. Fixed sides: top, bottom, left, or right side
3. Fixed port order: fixed sides and fixed order of ports on each side

Port Constraints

Four scenarios for constraints on port positions:

1. Free ports: arbitrary positions on the node's border
2. Fixed sides: top, bottom, left, or right side
3. Fixed port order: fixed sides and fixed order of ports on each side
4. Fixed ports: fixed exact positions on the node's border

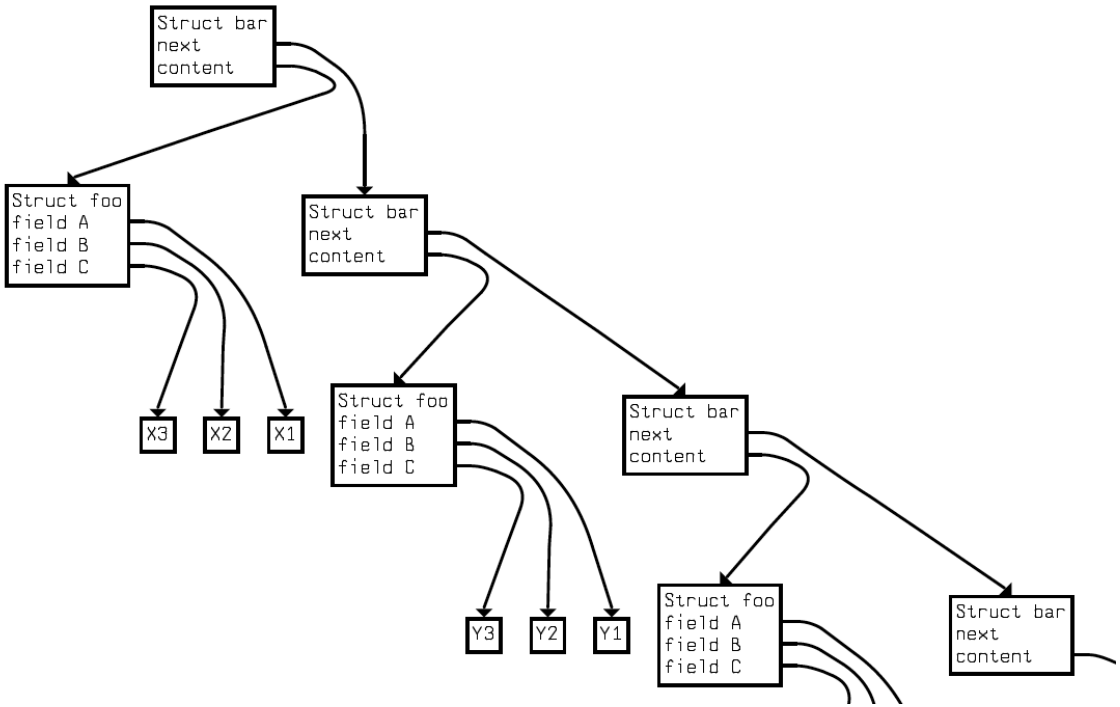
Port Constraints: Previous Work



E. R. Gansner, E. Koutsofios, S. C. North, K. Vo. *A technique for drawing directed graphs*, 1993.

- Gansner et al. 1993: port displacements for vertex placement

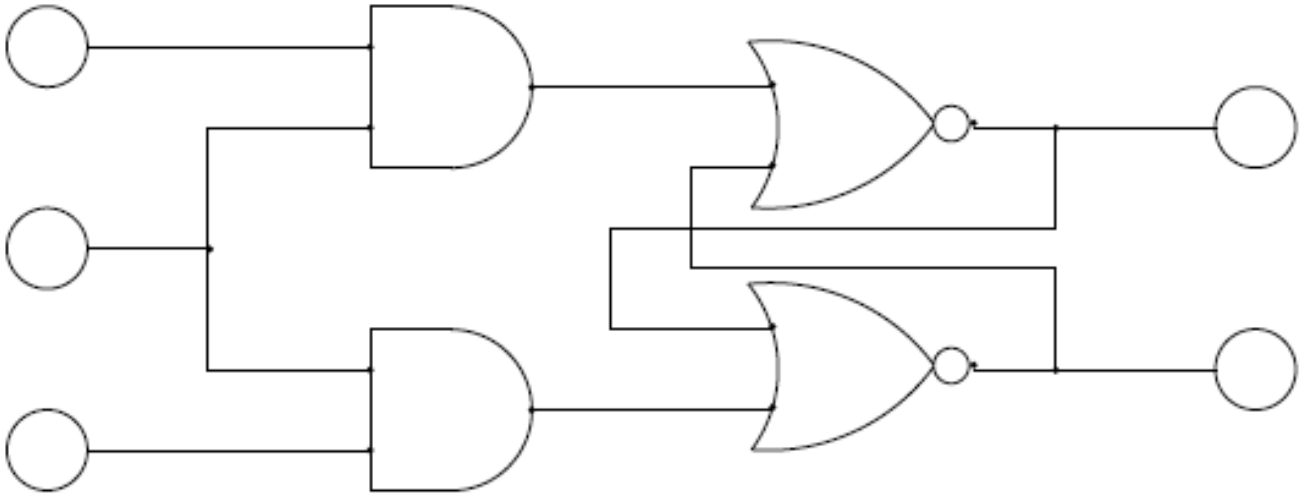
Port Constraints: Previous Work



G. Sander. *Graph layout through the VCG tool*, 1994.

- Gansner et al. 1993: port displacements for vertex placement
- Sander 1994: anchor points of edges

Port Constraints: Previous Work



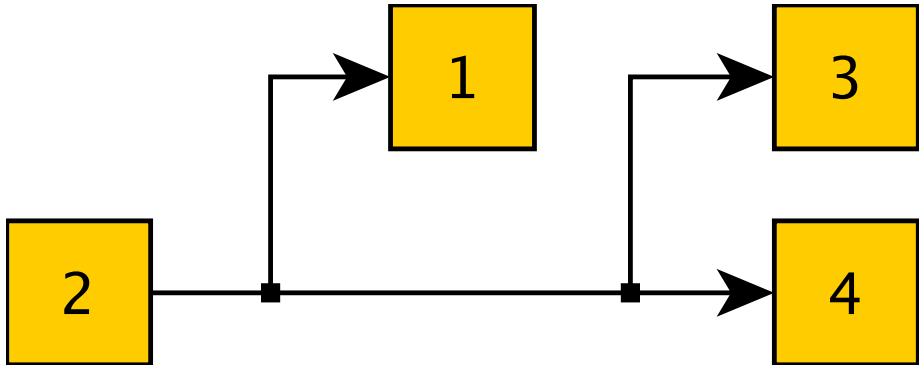
T. Eschbach. *Visualisierungen im Schaltkreisentwurf*, 2008.

- Gansner et al. 1993: port displacements for vertex placement
- Sander 1994: anchor points of edges
- Layout of circuit schematics: similar problem

Port Constraints: Previous Work

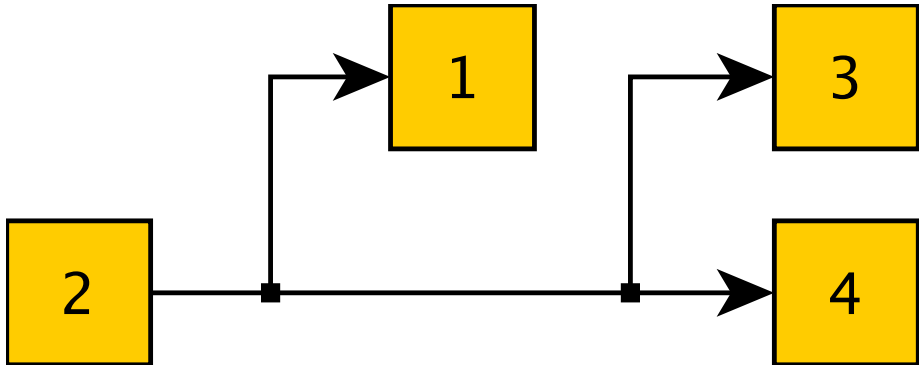
- Gansner et al. 1993: port displacements for vertex placement
- Sander 1994: anchor points of edges
- Layout of circuit schematics: similar problem
- Some commercial layout libraries support port constraints, no details published

Hyperedges



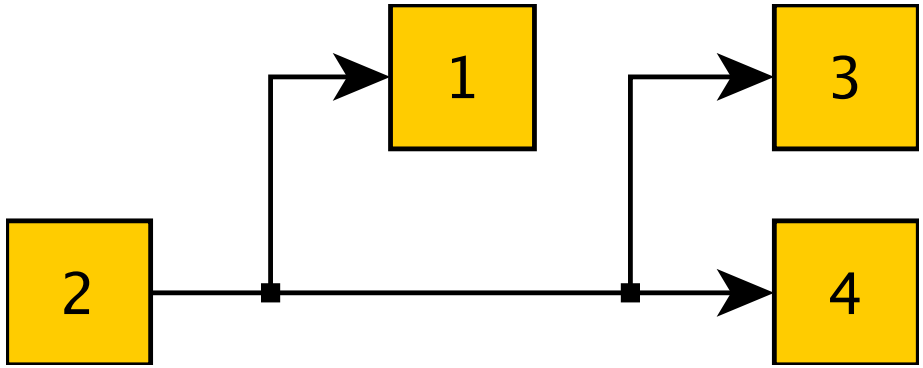
- One approach: handle hyperedges directly
  - Eschbach et al. 2006, Sander 2004

Hyperedges



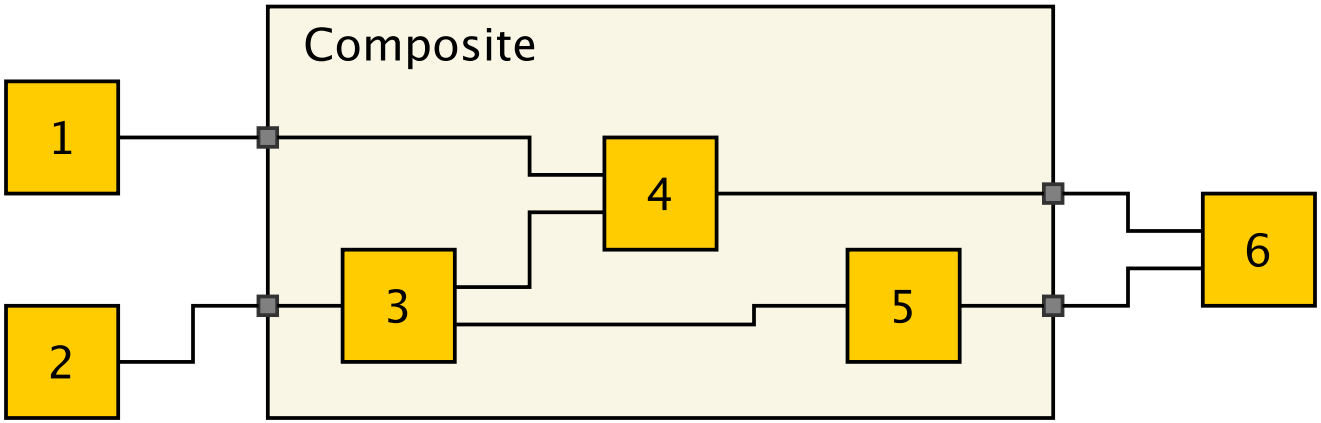
- One approach: handle hyperedges directly
  - Eschbach et al. 2006, Sander 2004
- Simpler approach: replace by a set of normal edges, e.g. $(2, 1)$, $(2, 3)$, $(2, 4)$

Hyperedges



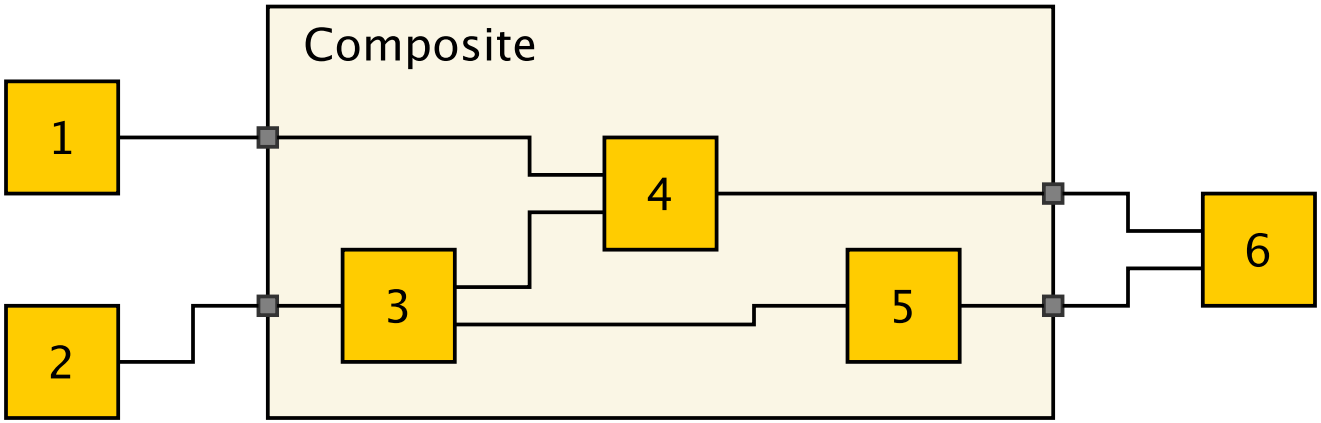
- One approach: handle hyperedges directly
  - Eschbach et al. 2006, Sander 2004
- Simpler approach: replace by a set of normal edges, e.g. $(2, 1)$, $(2, 3)$, $(2, 4)$
- Not unique for multiple sources *and* multiple targets, but OK for most data flow diagrams

Compound Diagrams



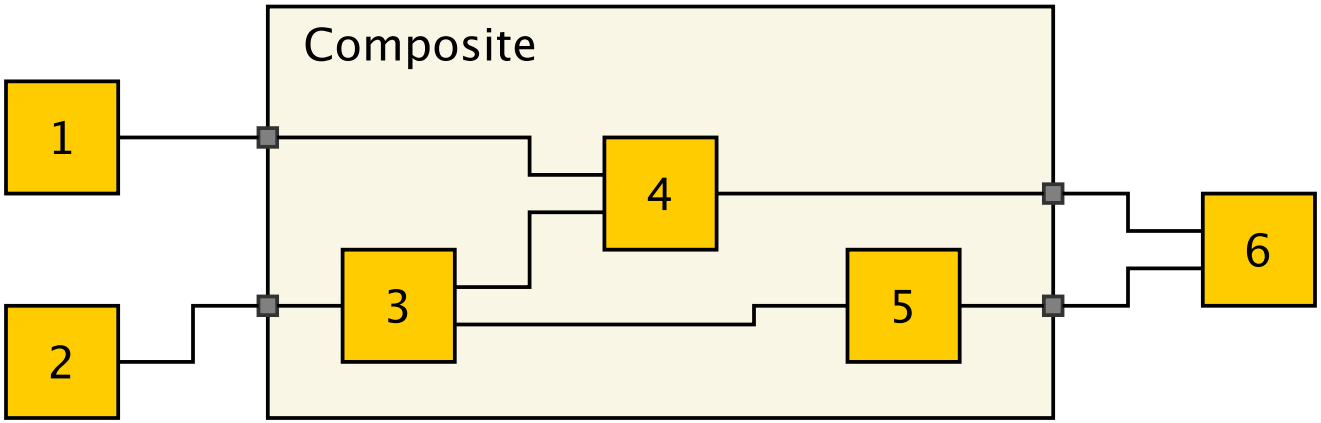
- Compound operators may contain nested diagrams

Compound Diagrams



- Compound operators may contain nested diagrams
- There are approaches to handle general compound graphs
  - Sugiyama et al. 1991, Sander 1996

Compound Diagrams



- Compound operators may contain nested diagrams
- There are approaches to handle general compound graphs
  - Sugiyama et al. 1991, Sander 1996
- Different situation: edges do not cross vertex boundaries, but may be connected with external ports

1. Introduction – Problem Statement and Related Work
   - Data Flow Diagrams
   - Port Constraints
   - Hyperedges
   - Compound Diagrams

2. Extensions of Hierarchical Layout
   - Assignment of Dummy Vertices
   - Crossing Minimization
   - Orthogonal Edge Routing

3. Results
   - Layout Gallery
   - Applications and Demonstration

The Hierarchical Approach

Steps of our variant of the hierarchical layout algorithm:

1. Cycle removal
2. Layer assignment
3. Crossing minimization
4. Edge routing I
5. Vertex placement
6. Edge routing II

The Hierarchical Approach

Steps of our variant of the hierarchical layout algorithm:

1. Cycle removal
2. **Layer assignment**
3. **Crossing minimization**
4. **Edge routing I**
5. Vertex placement
6. Edge routing II

- Main contributions are in the highlighted steps

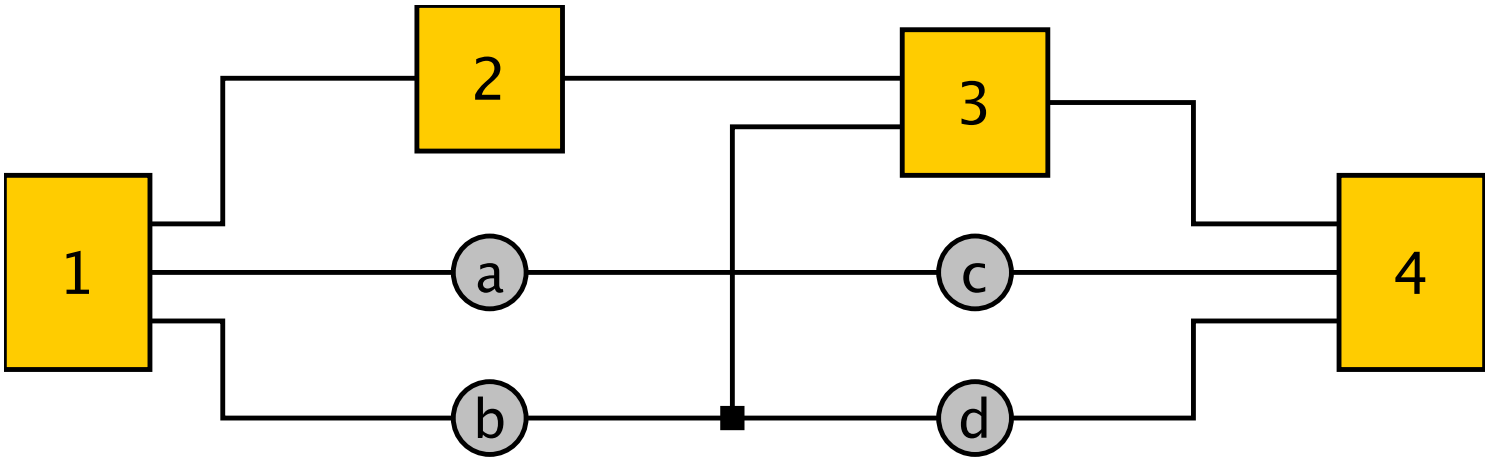
The Hierarchical Approach

Steps of our variant of the hierarchical layout algorithm:

1. Cycle removal
2. **Layer assignment**
3. **Crossing minimization**
4. **Edge routing I**
5. Vertex placement
6. Edge routing II

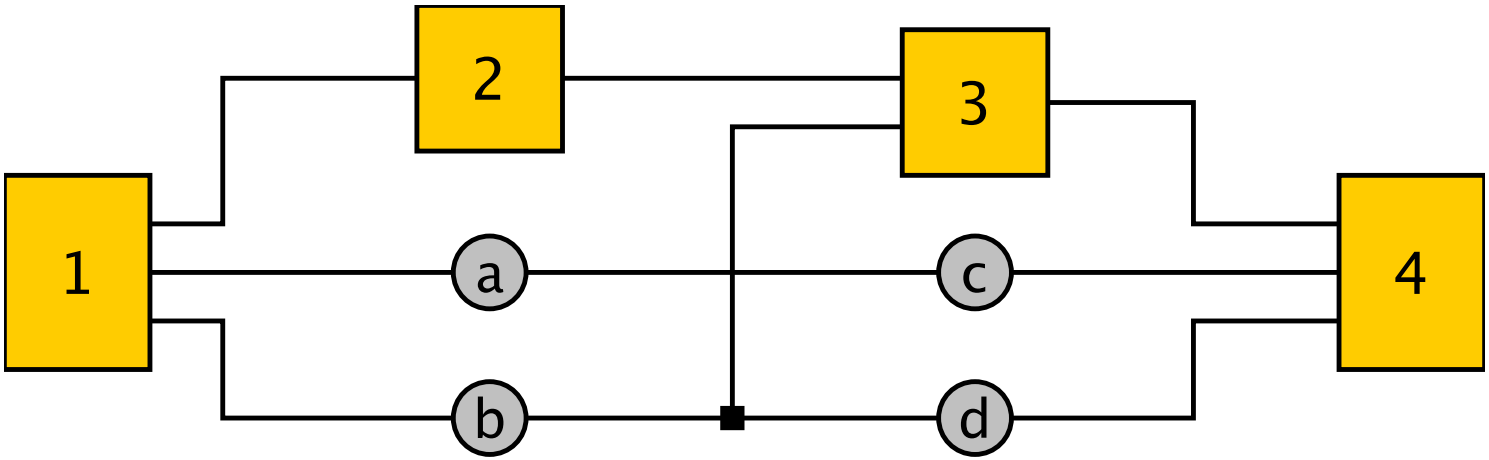
- Main contributions are in the highlighted steps
- Standard methods for other steps
  - G. Di Battista, P. Eades, R. Tamassia, I. G. Tollis 1999
  - Sander 1996, 2004

Step 2: Assignment of Dummy Vertices



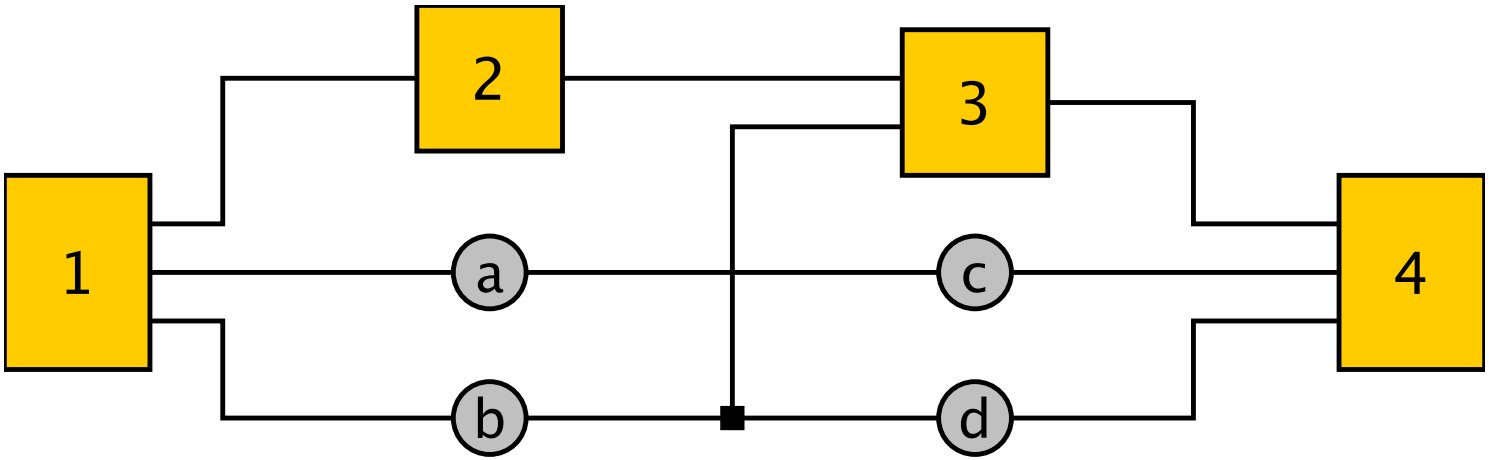
- A standard method is used for assigning vertices to layers

Step 2: Assignment of Dummy Vertices



- A standard method is used for assigning vertices to layers
- Dummy vertices are used to split long edges

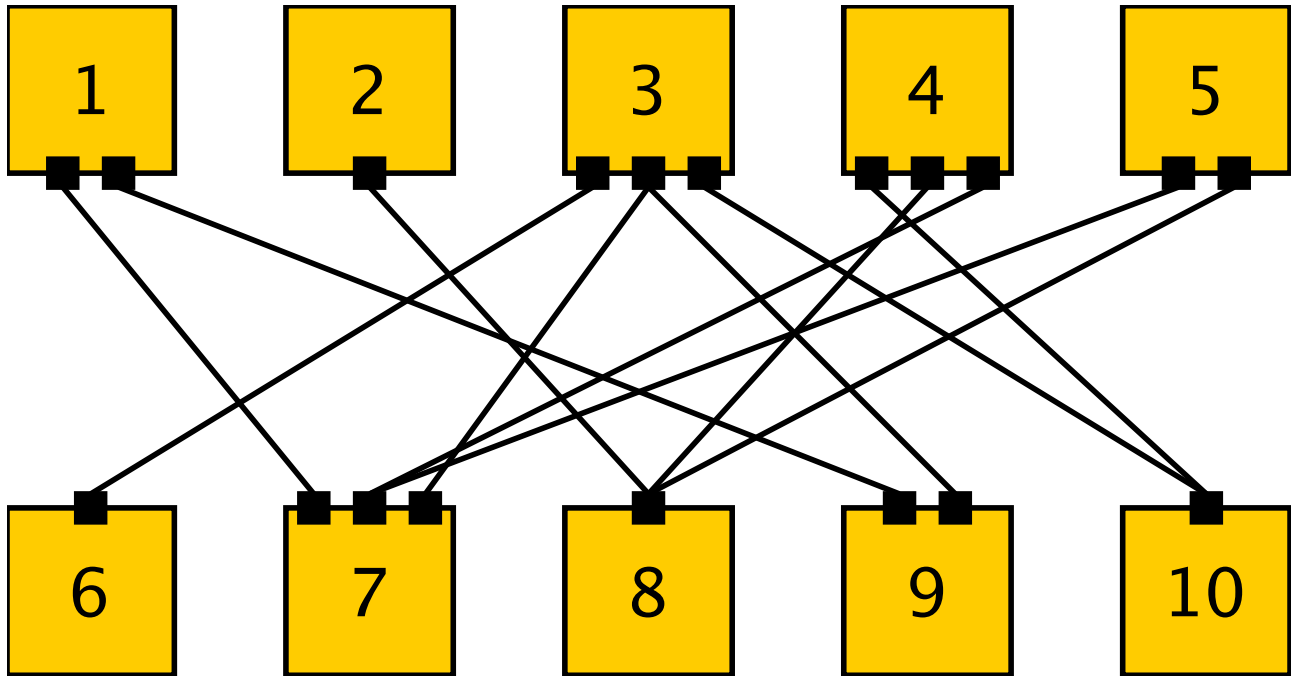
Step 2: Assignment of Dummy Vertices



- A standard method is used for assigning vertices to layers
- Dummy vertices are used to split long edges
- Merge dummy vertices of long edges that belong to the same hyperedge
  - Edges $(1, 3)$ and $(1, 4)$ share the dummy vertex b

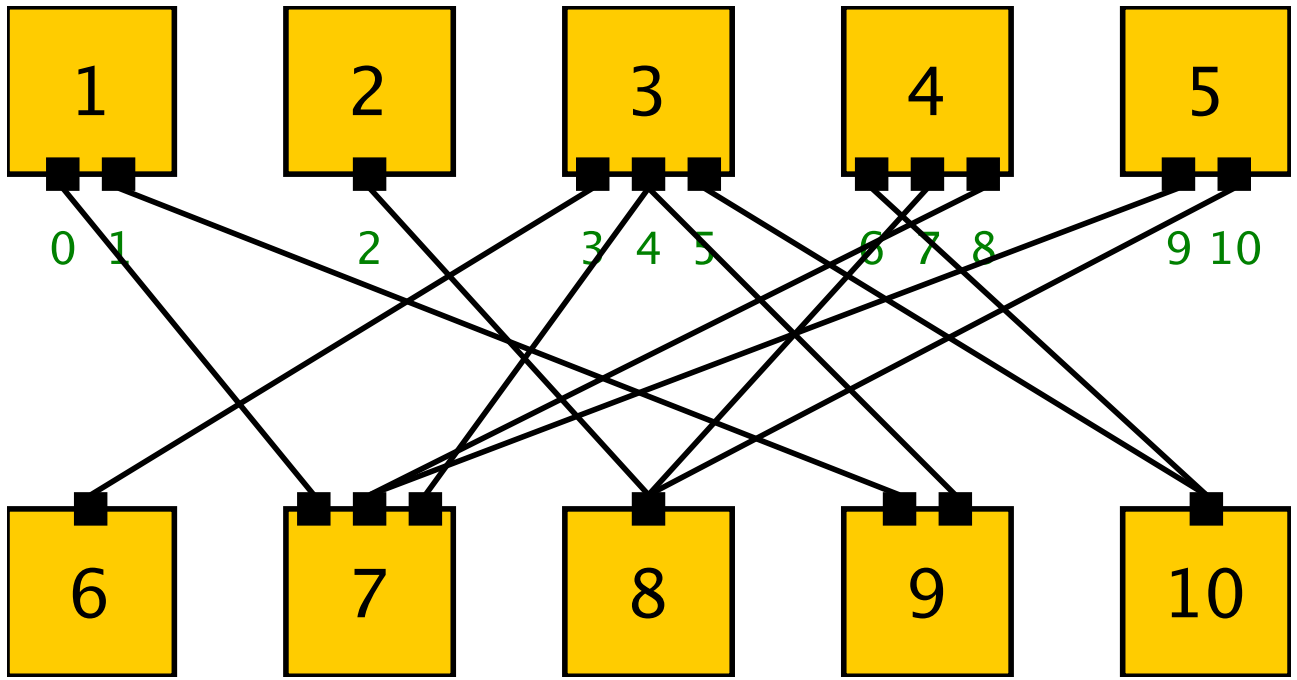
Step 3: Crossing Minimization

- Use two-layer crossing minimization method for layer-by-layer sweep



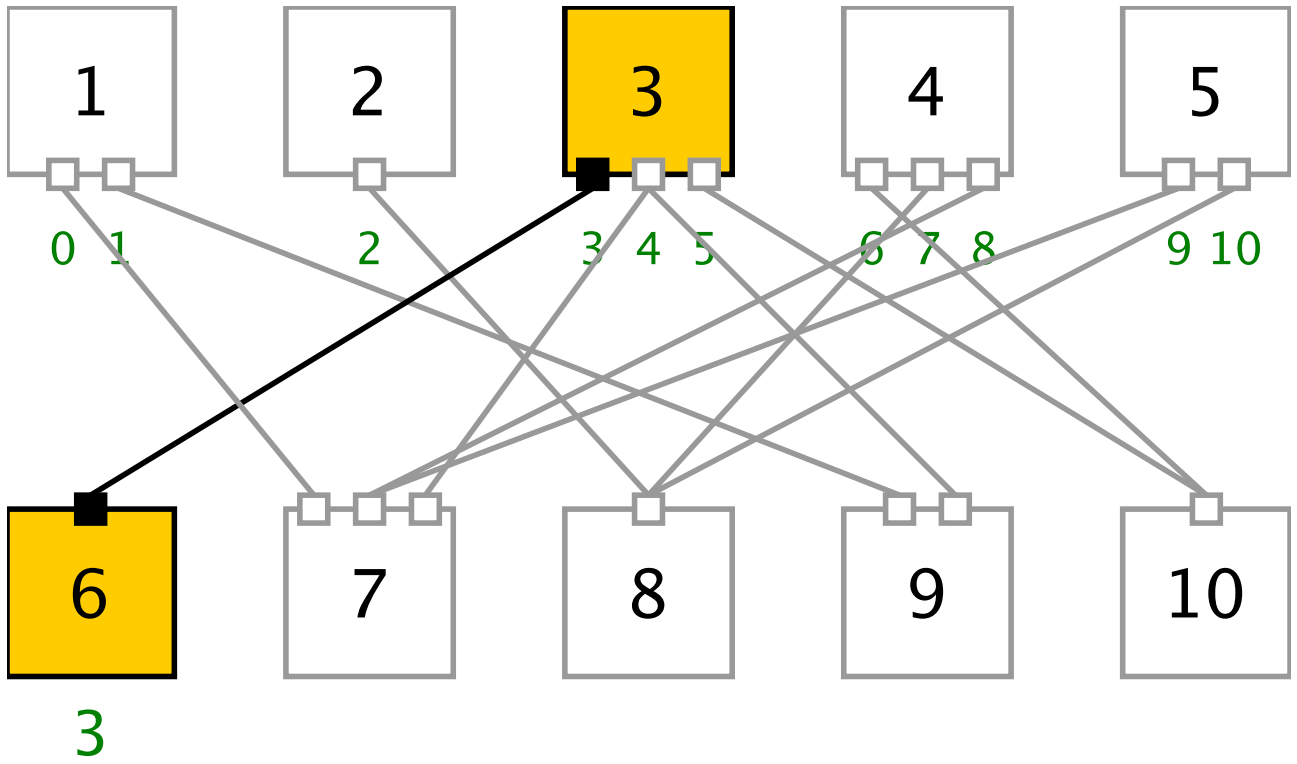
Step 3: Crossing Minimization: Barycenter Method

- Extended barycenter method: determine port ranks in the fixed layer



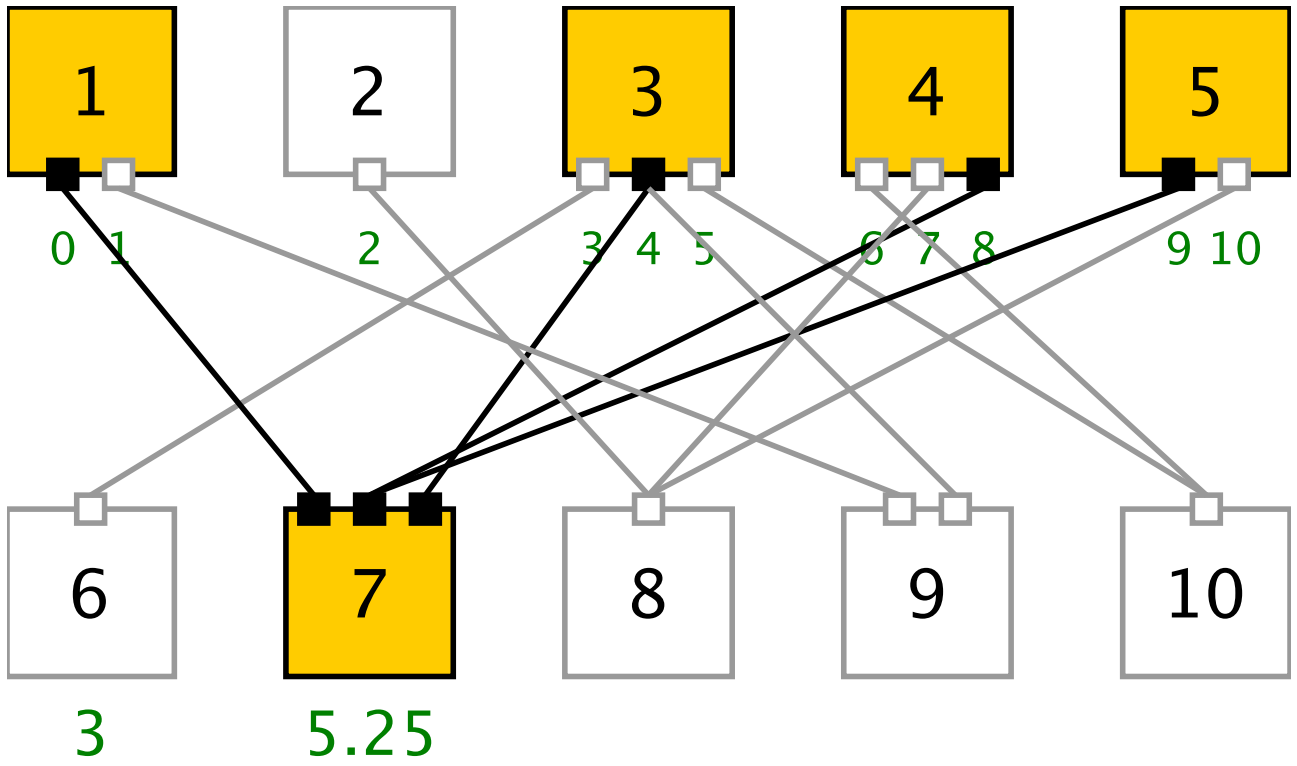
Step 3: Crossing Minimization: Barycenter Method

- Calculate barycenter values as average of the ranks of connected ports for all nodes in the free layer



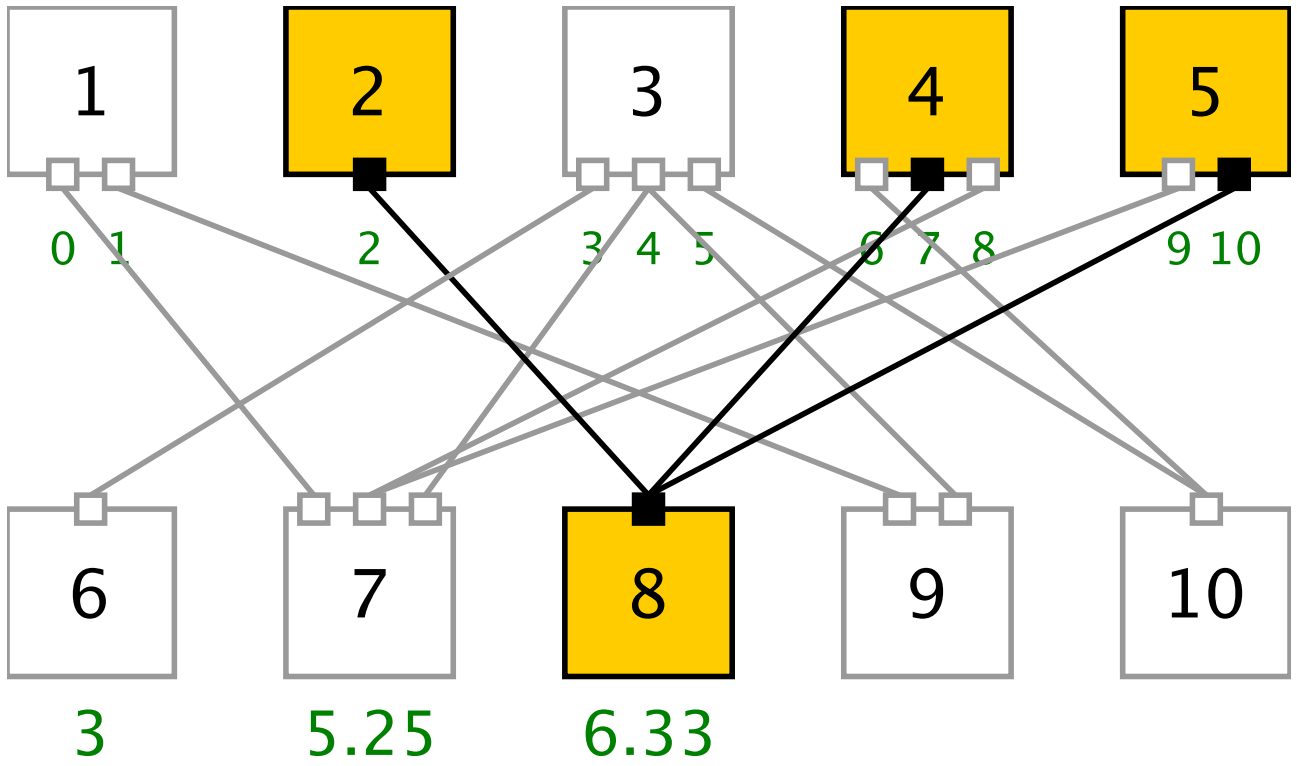
Step 3: Crossing Minimization: Barycenter Method

- Calculate barycenter values as average of the ranks of connected ports for all nodes in the free layer



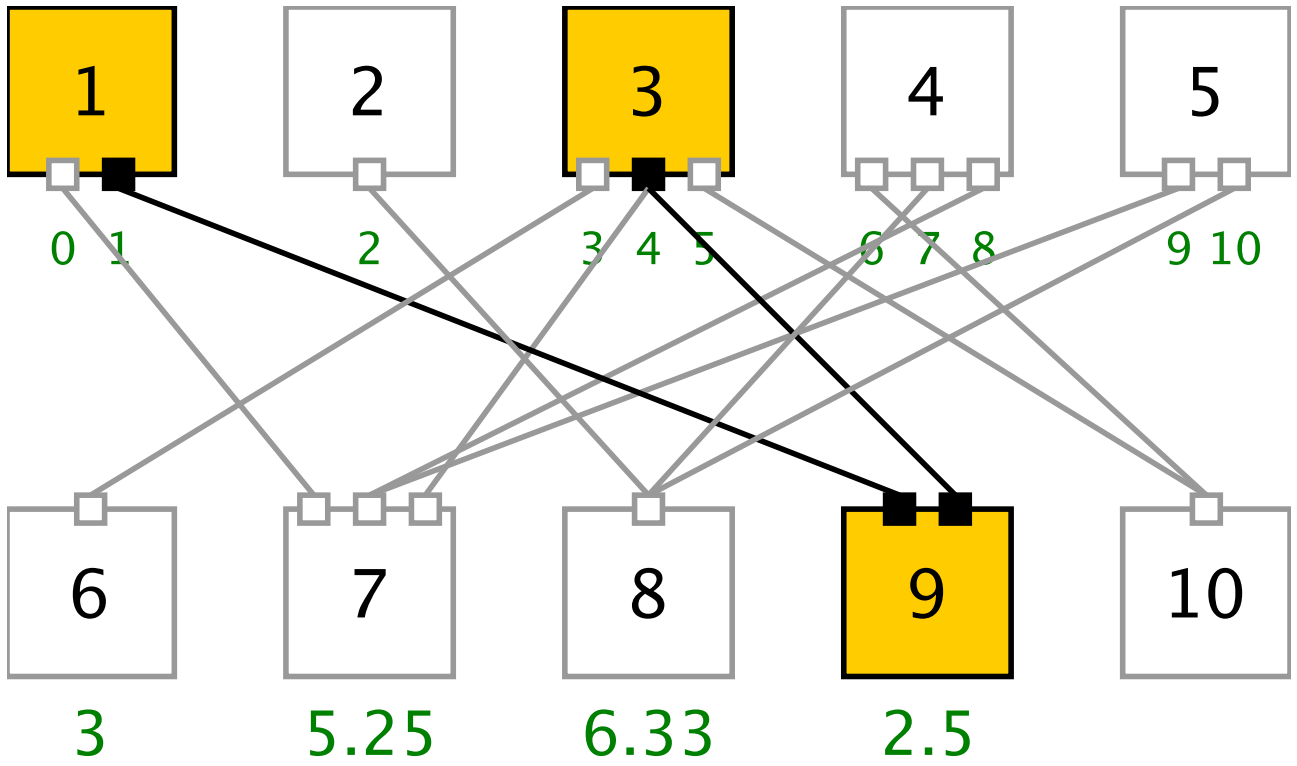
Step 3: Crossing Minimization: Barycenter Method

- Calculate barycenter values as average of the ranks of connected ports for all nodes in the free layer



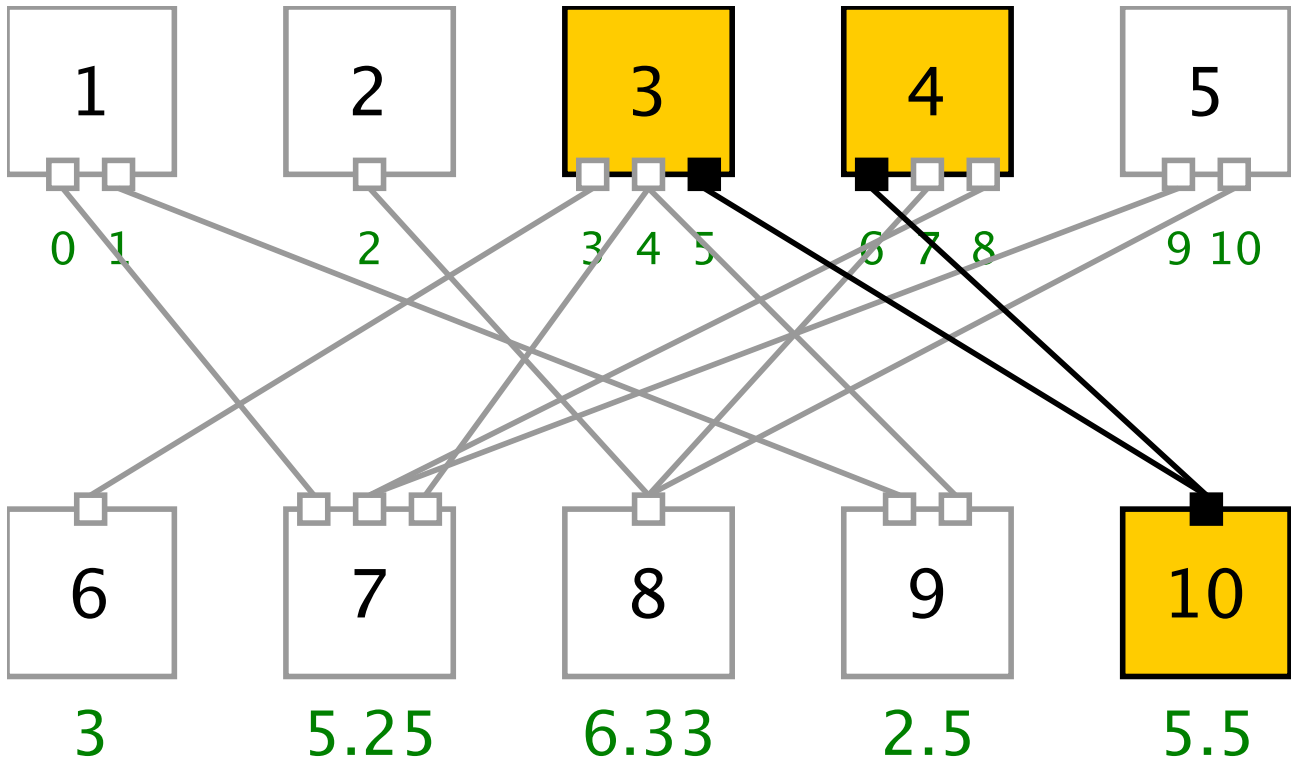
Step 3: Crossing Minimization: Barycenter Method

- Calculate barycenter values as average of the ranks of connected ports for all nodes in the free layer



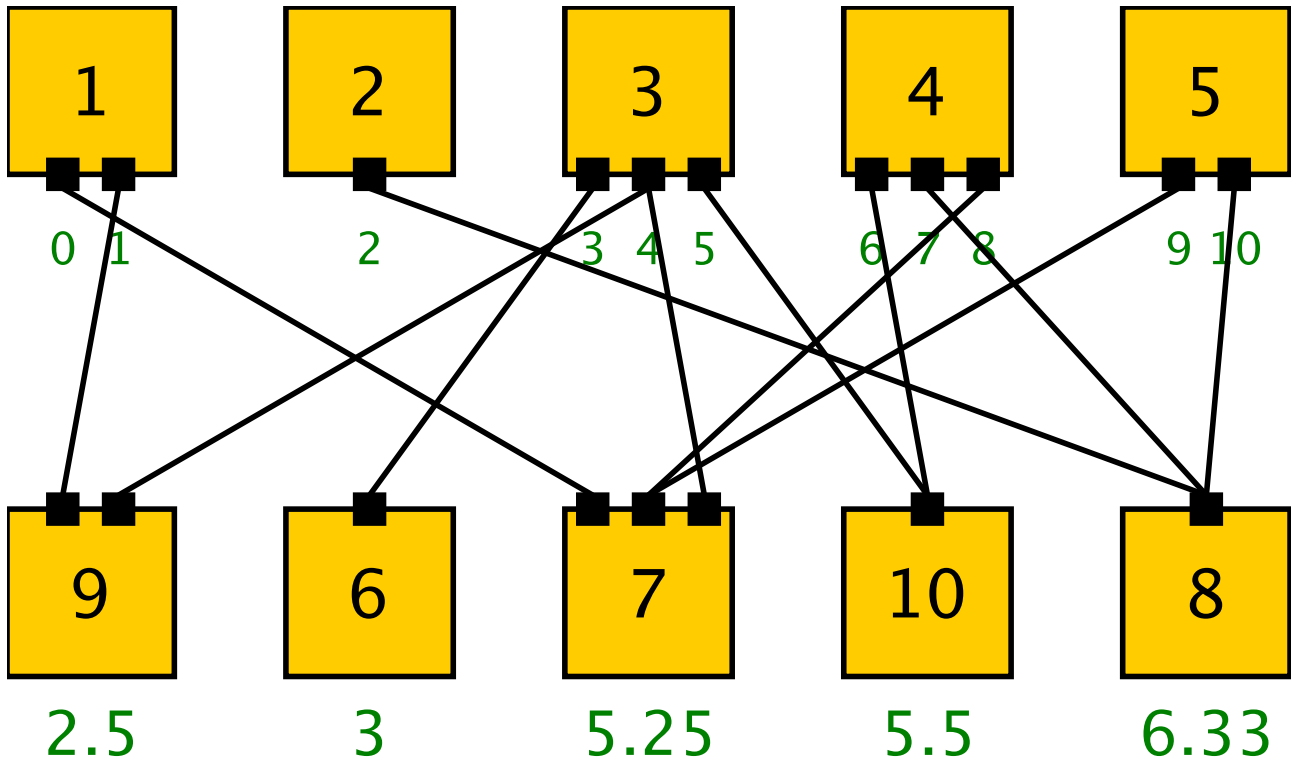
Step 3: Crossing Minimization: Barycenter Method

- Calculate barycenter values as average of the ranks of connected ports for all nodes in the free layer

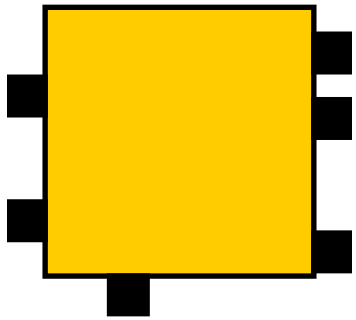


Step 3: Crossing Minimization: Barycenter Method

- Order the nodes in the free layer by their barycenter values

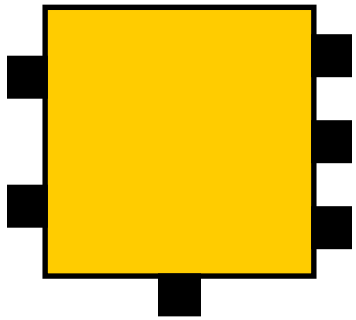


Scenarios of Port Constraints



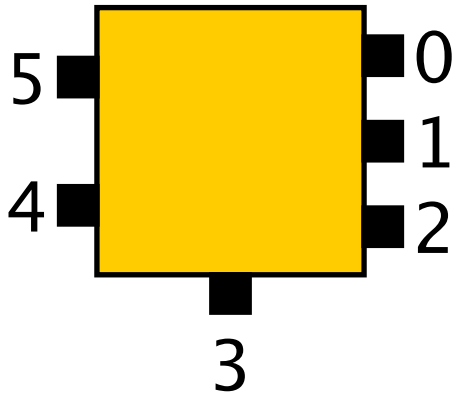
- Fixed ports: use extended barycenter method

Scenarios of Port Constraints



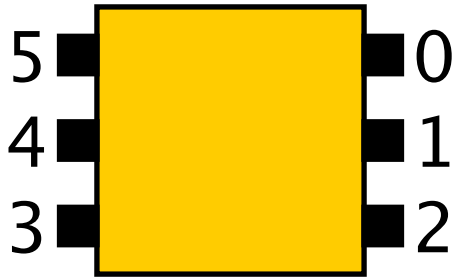
- Fixed ports: use extended barycenter method
- Fixed port order: as with fixed ports; distribute ports evenly on the vertex border first

Scenarios of Port Constraints



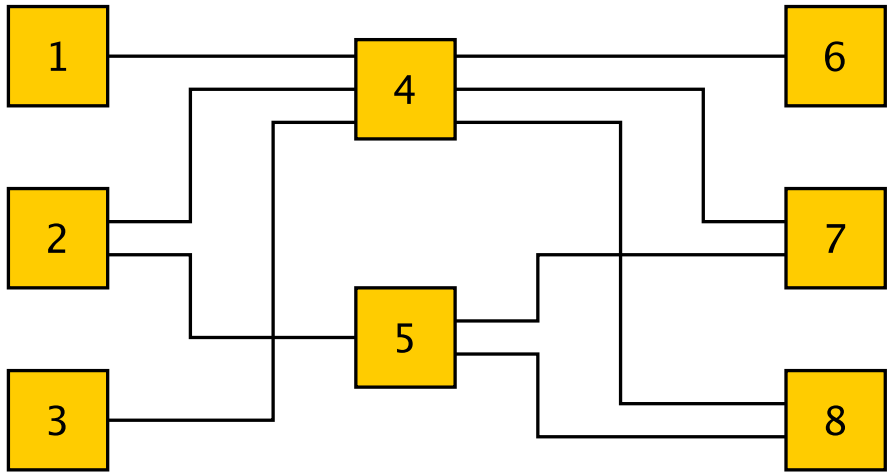
- Fixed ports: use extended barycenter method
- Fixed port order: as with fixed ports; distribute ports evenly on the vertex border first
- Fixed sides: set order of ports for each vertex using the extended barycenter method
  - Determine vertex barycenters from their port barycenters

Scenarios of Port Constraints



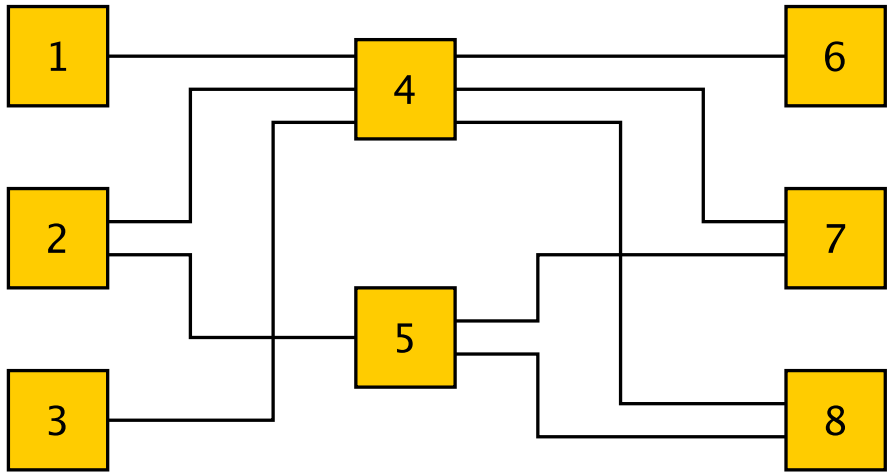
- Fixed ports: use extended barycenter method
- Fixed port order: as with fixed ports; distribute ports evenly on the vertex border first
- Fixed sides: set order of ports for each vertex using the extended barycenter method
  - Determine vertex barycenters from their port barycenters
- Free ports: as with fixed sides; first put input ports left, output ports right

Orthogonal Edge Routing



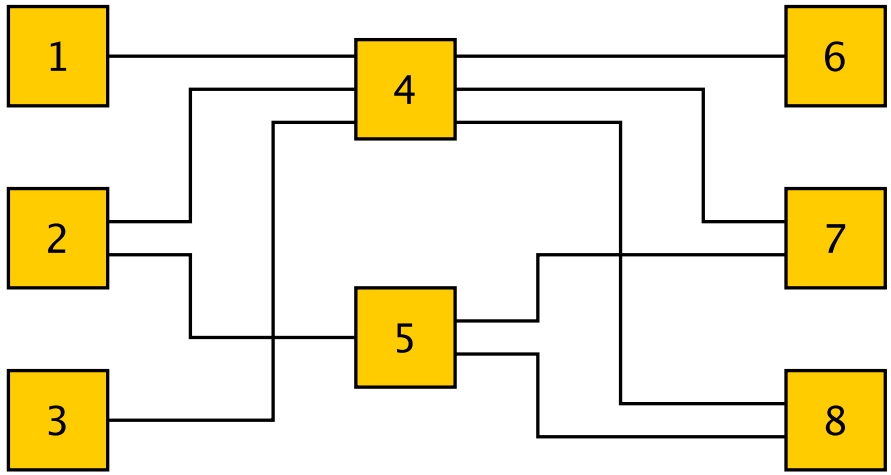
- Create appropriate horizontal and vertical line segments

Orthogonal Edge Routing



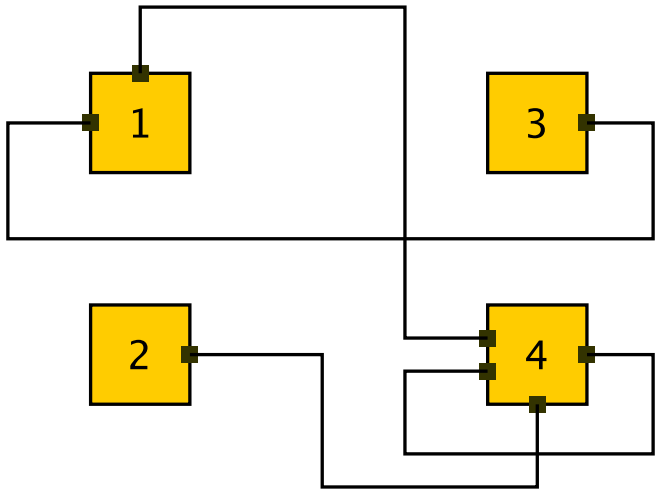
- Create appropriate horizontal and vertical line segments
- Can apply standard methods if the source port is on the right side and target port is on the left side
  - Eschbach et al. 2006, Sander 1996, Baburin 2002

Orthogonal Edge Routing



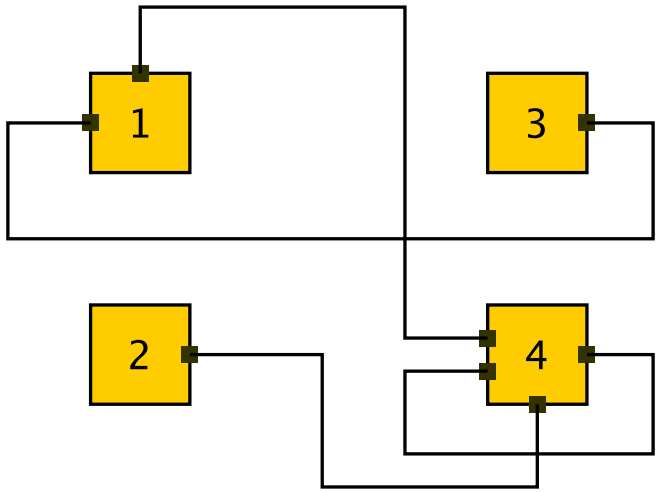
- Create appropriate horizontal and vertical line segments
- Can apply standard methods if the source port is on the right side and target port is on the left side
  - Eschbach et al. 2006, Sander 1996, Baburin 2002
- What about ports on other sides?

Step 4: Routing Around Vertices



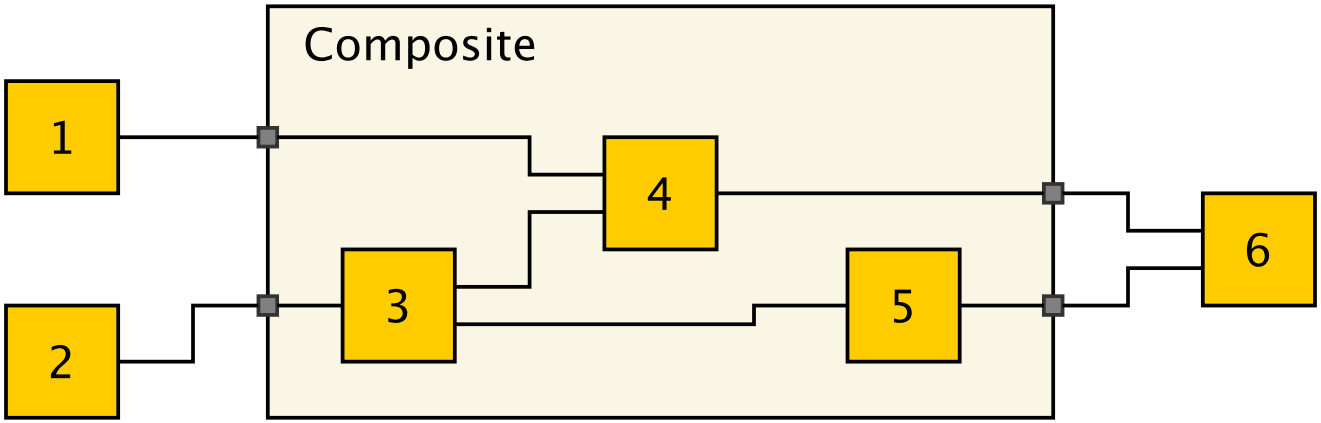
- Need to route edges around vertices

Step 4: Routing Around Vertices



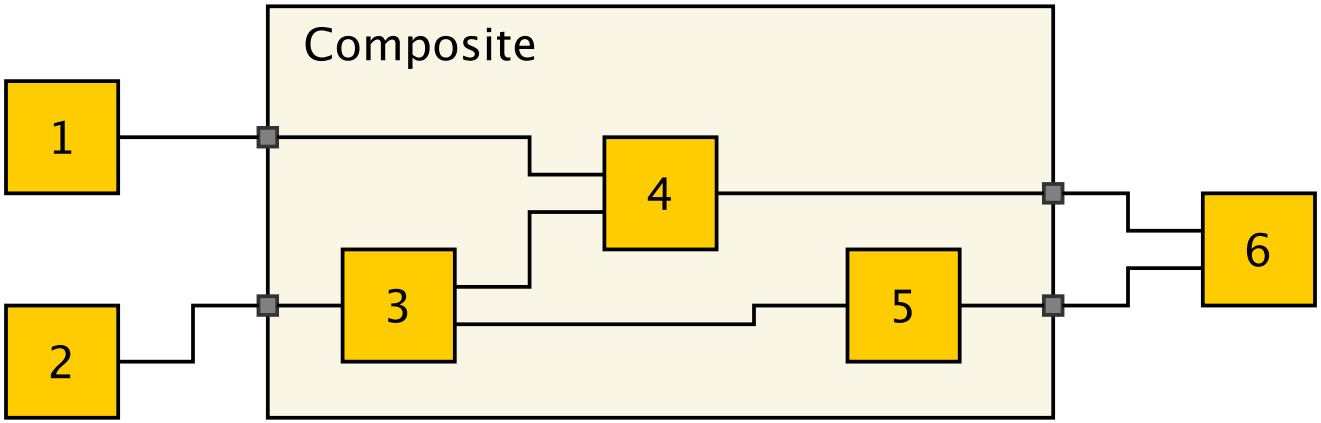
- Need to route edges around vertices
- Apply additional routing step after crossing minimization
  - Order the edges appropriately
  - Determine amount of space needed for the edges, use this information in the subsequent vertex placement step

Step 6: Routing to External Ports



- If a compound diagram has connections to external ports, they must be properly routed

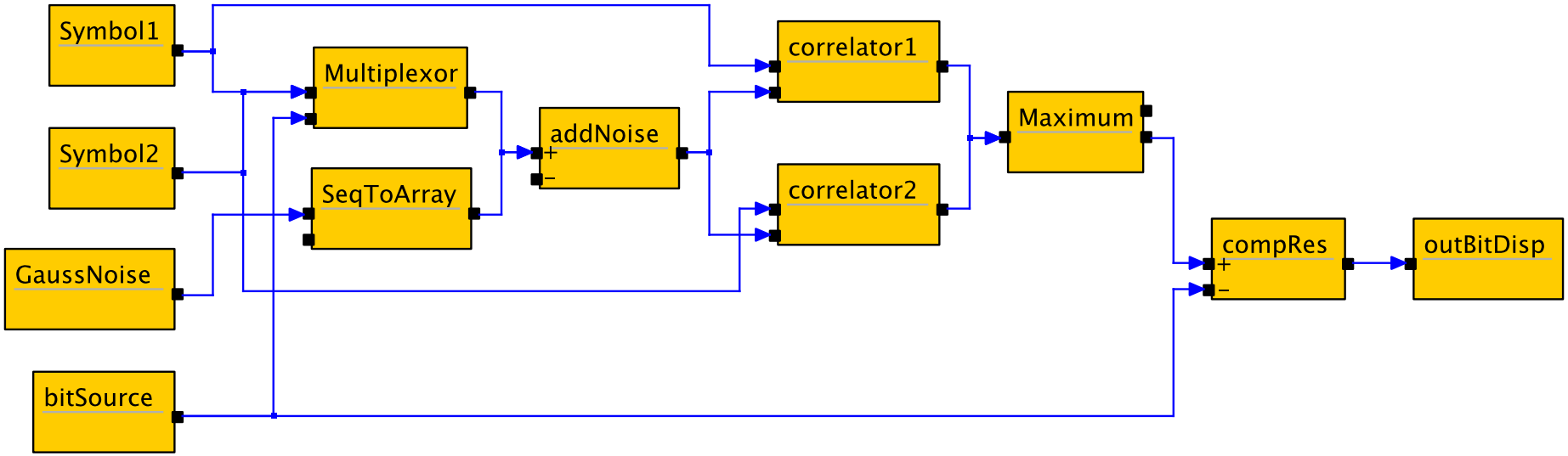
Step 6: Routing to External Ports



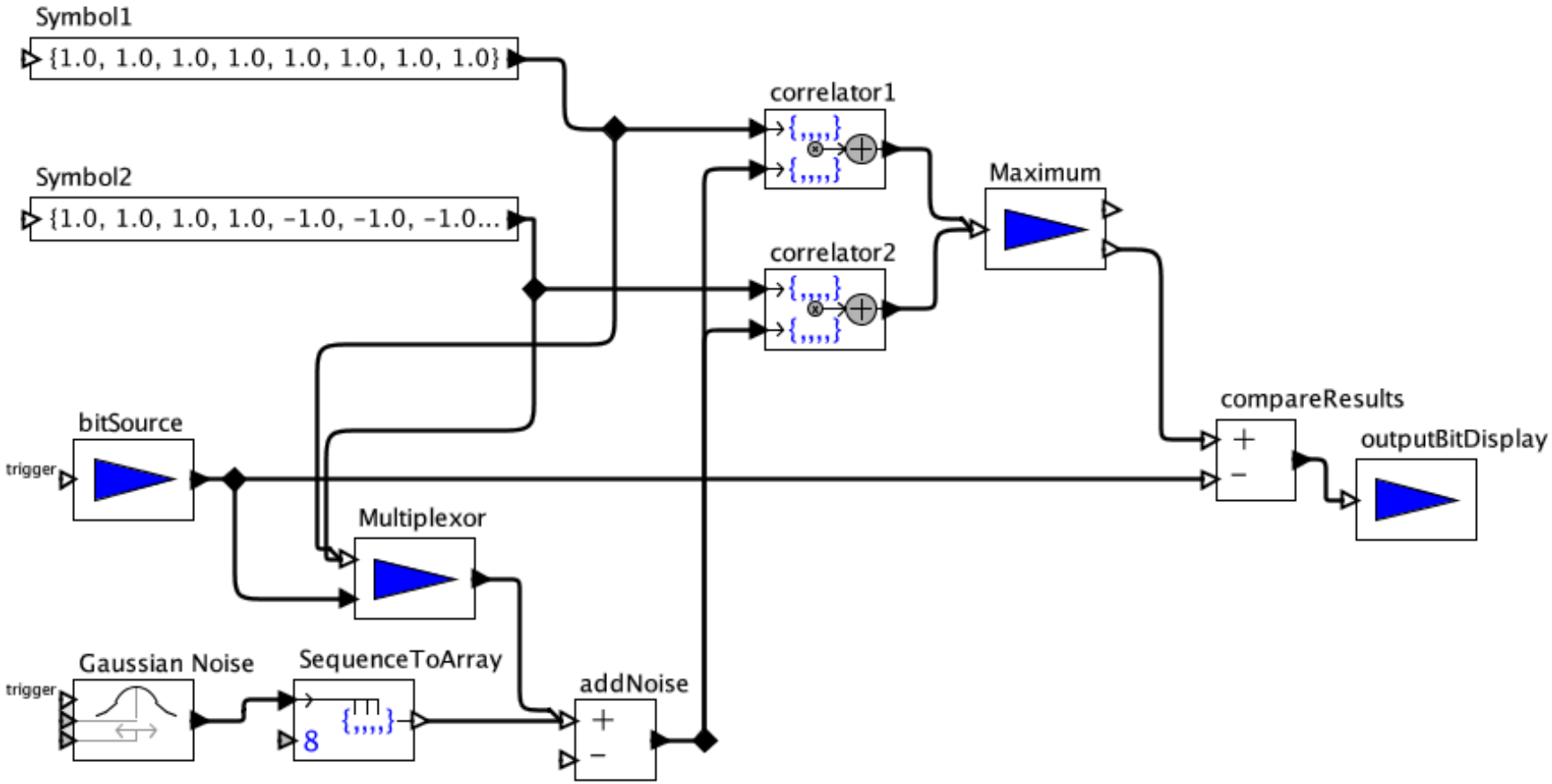
- If a compound diagram has connections to external ports, they must be properly routed
- General approach: apply layout recursively
  - Treat external ports as ordinary vertices in most steps of the algorithm
  - Input ports to the first layer, output ports to the last layer
  - Need special handling to place ports on the border of the parent vertex and route connected edges

1. Introduction – Problem Statement and Related Work
   - Data Flow Diagrams
   - Port Constraints
   - Hyperedges
   - Compound Diagrams
2. Extensions of Hierarchical Layout
   - Assignment of Dummy Vertices
   - Crossing Minimization
   - Orthogonal Edge Routing
3. Results
   - Layout Gallery
   - Applications and Demonstration

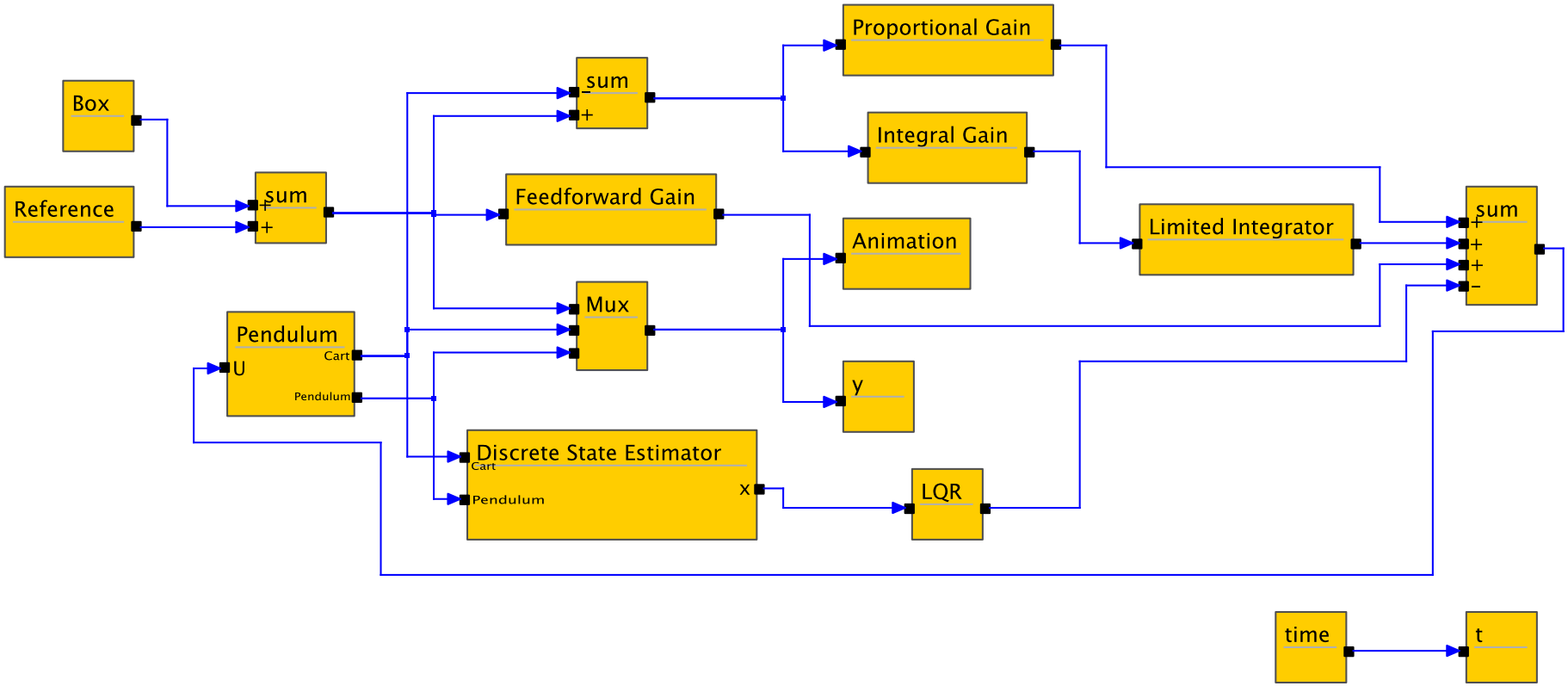
Layout Gallery (1/5)



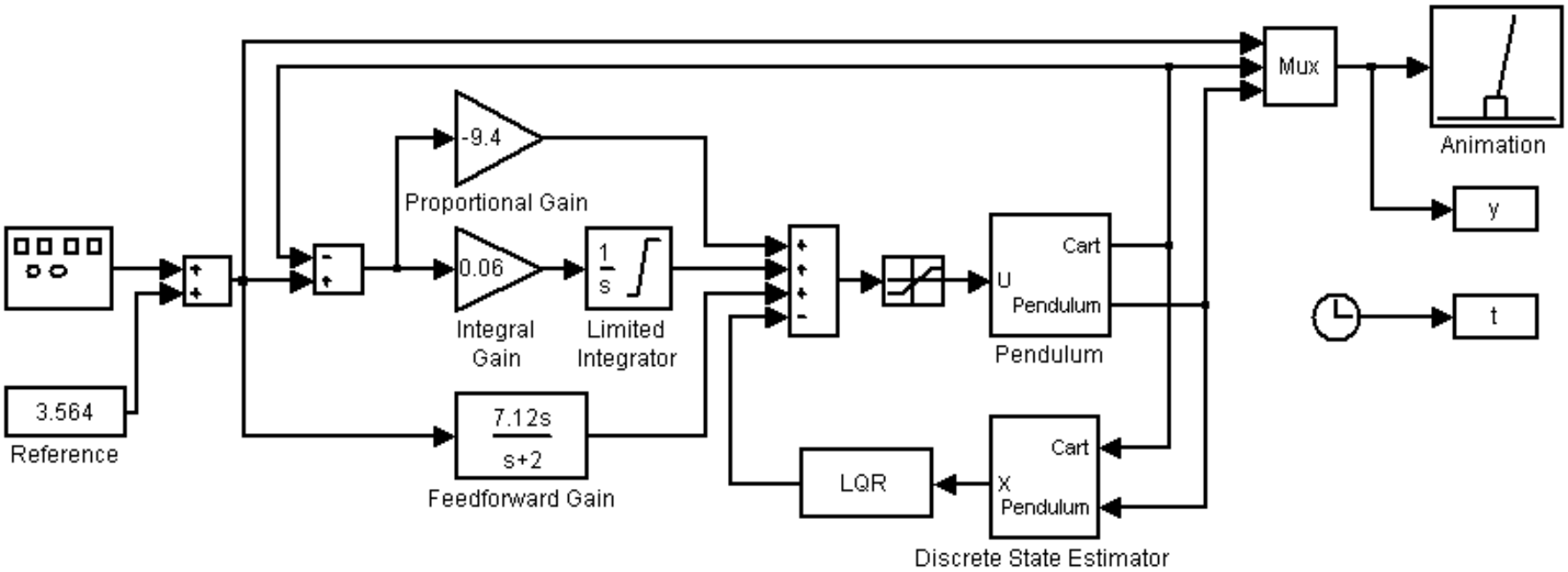
Layout Gallery (1/5): Original Layout



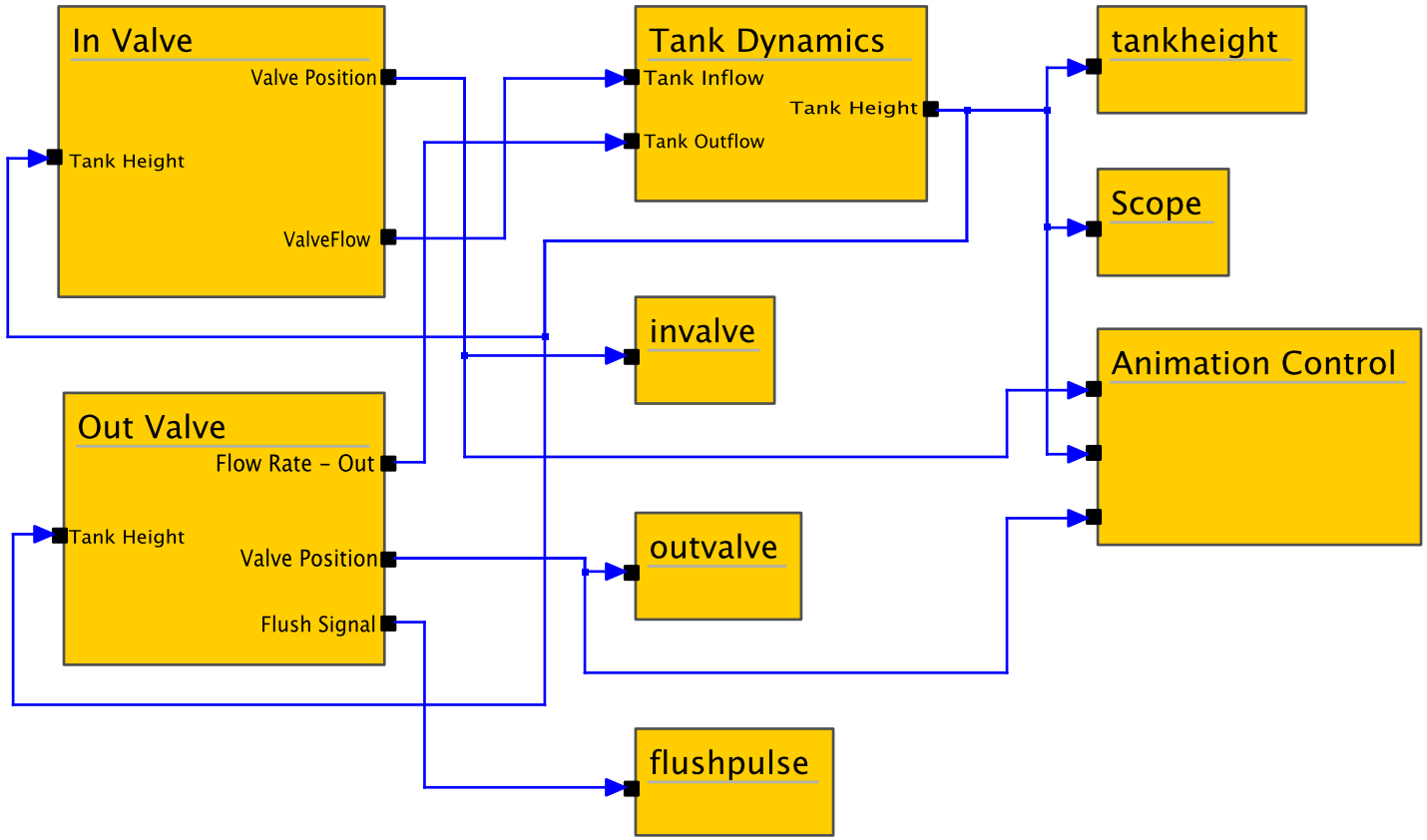
Layout Gallery (2/5)



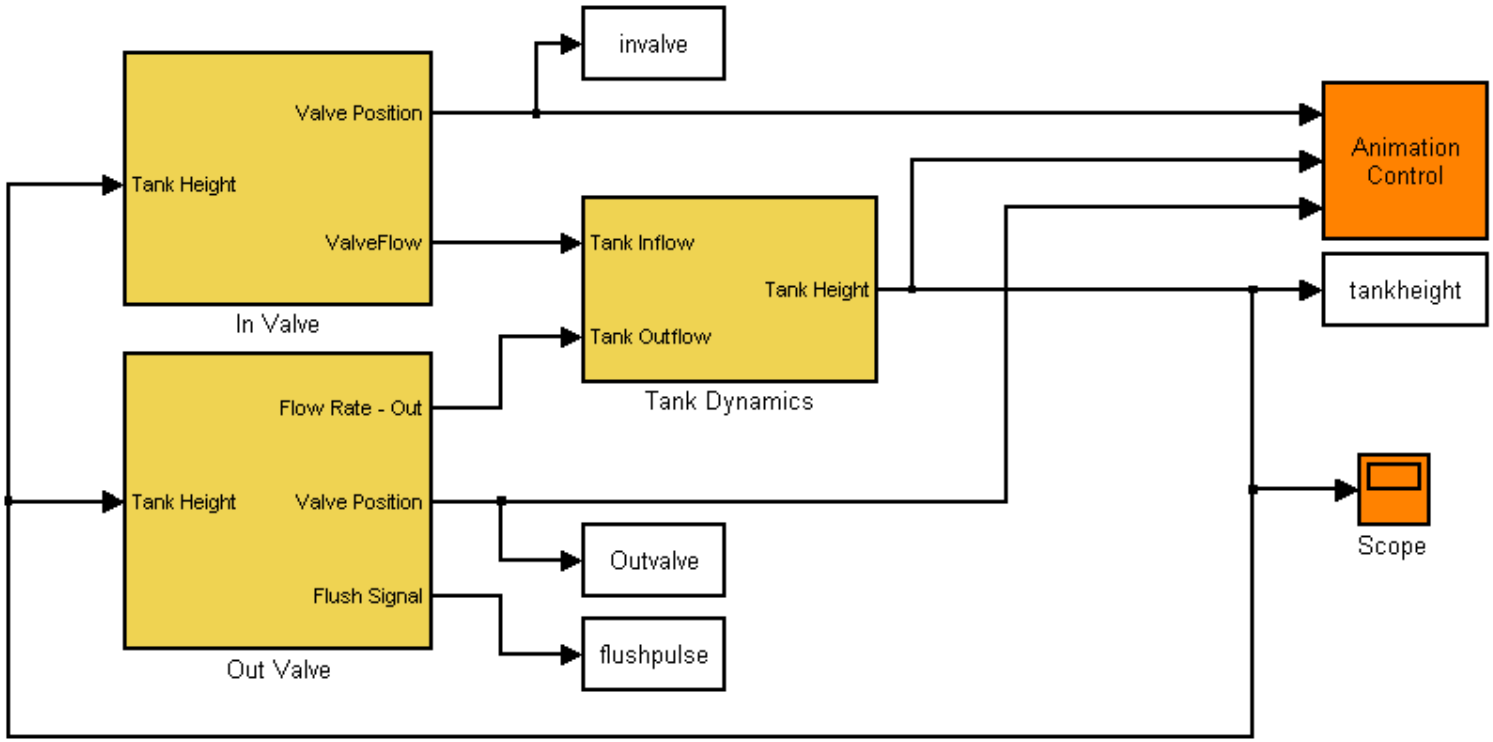
Layout Gallery (2/5): Original Layout



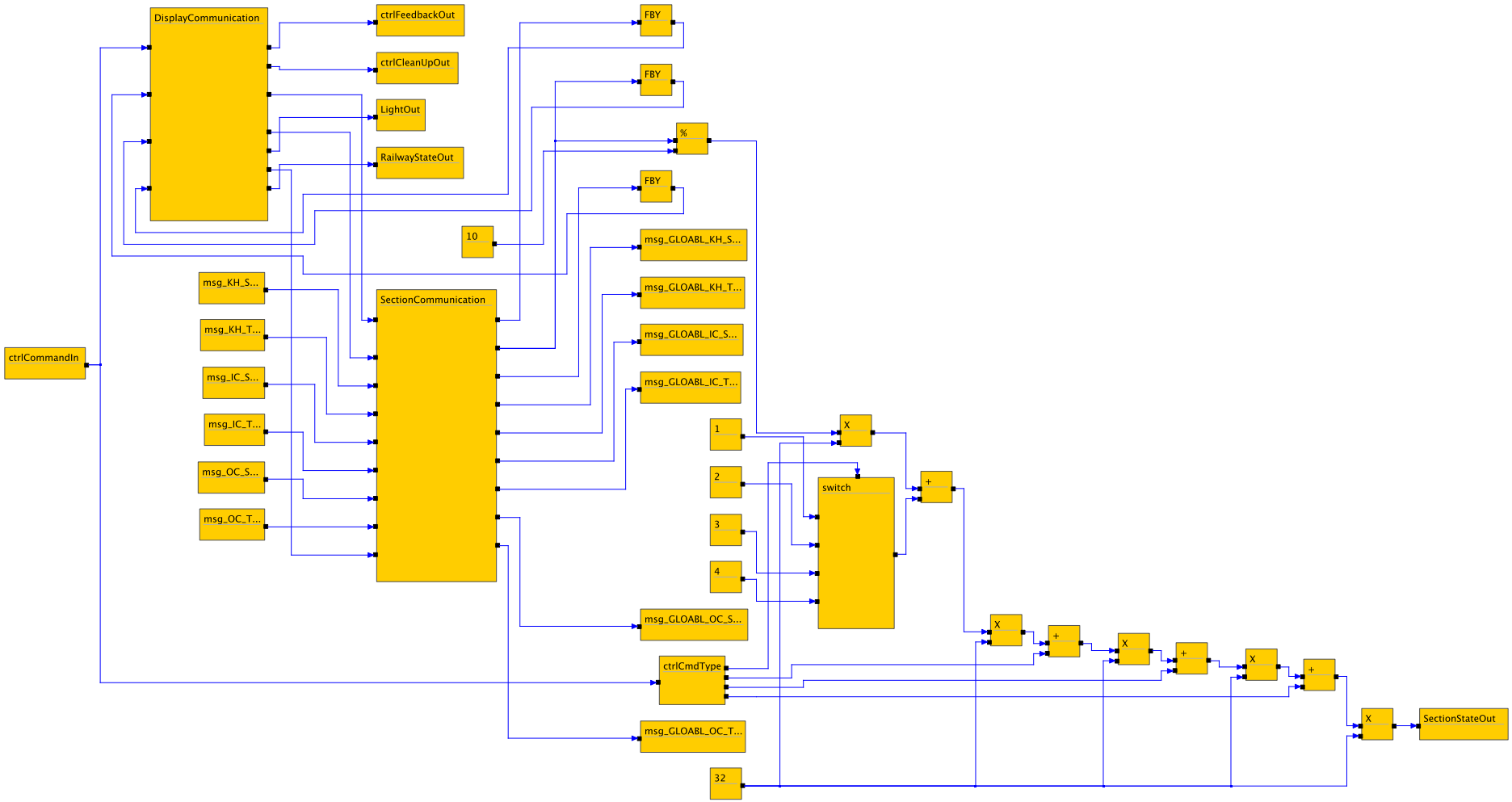
Layout Gallery (3/5)



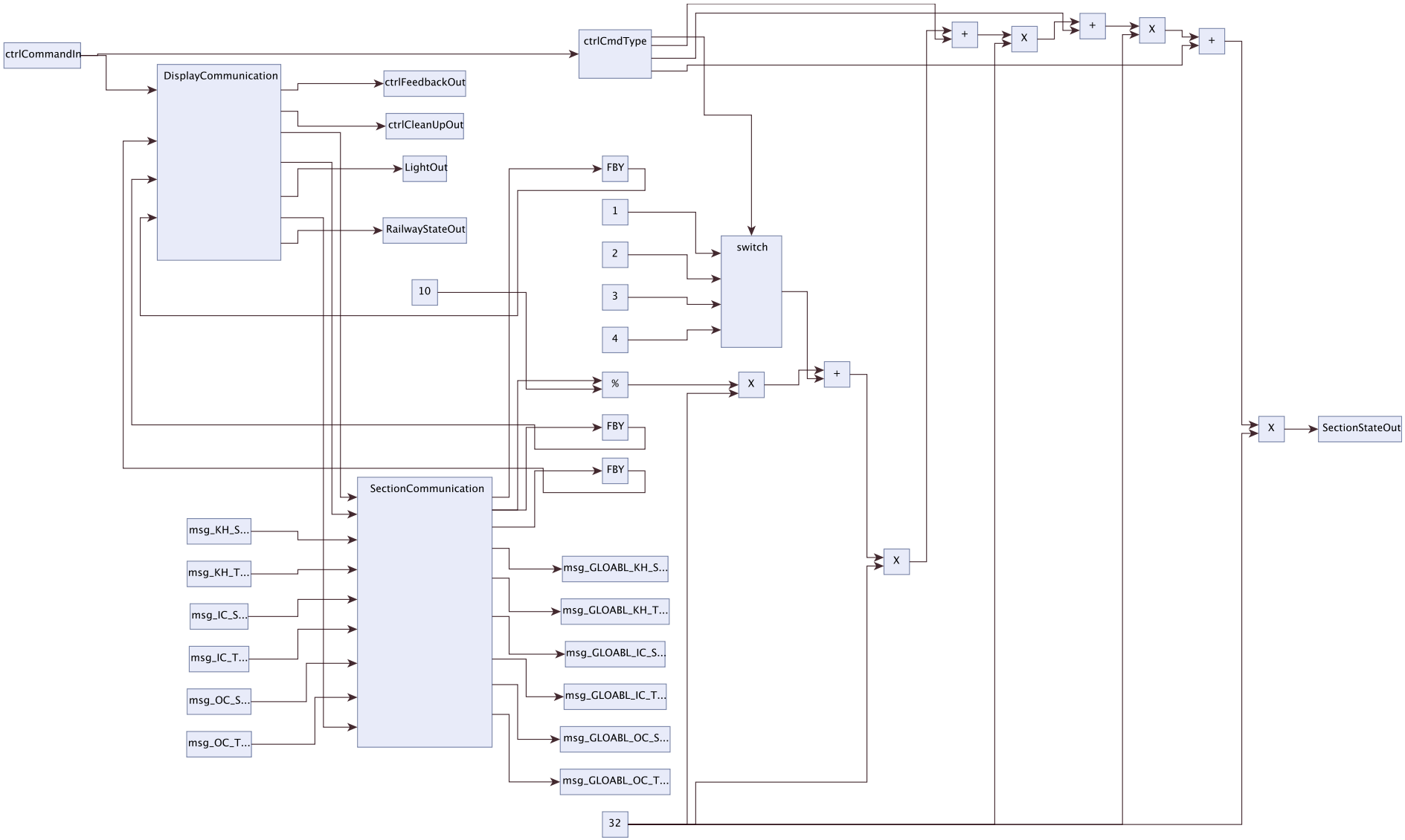
Layout Gallery (3/5): Original Layout



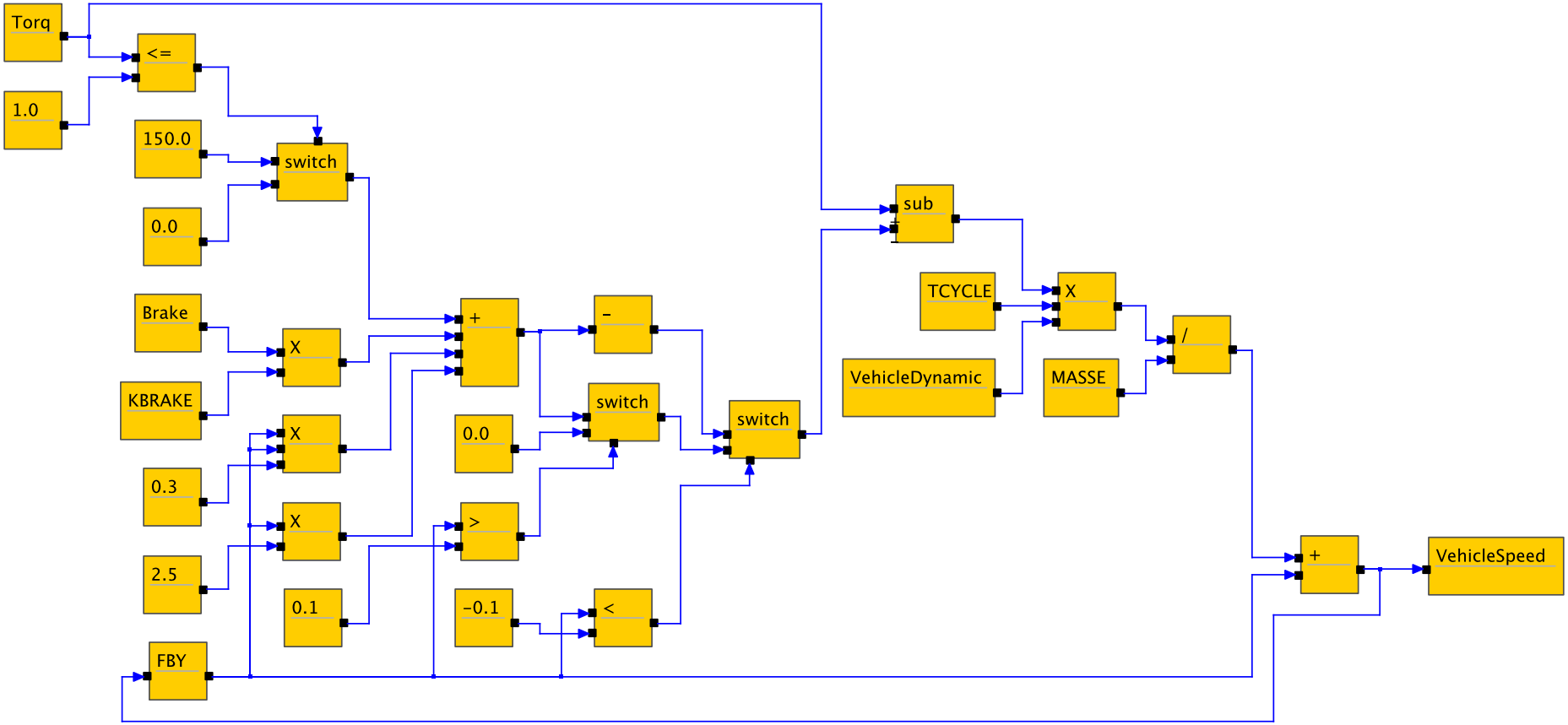
Layout Gallery (4/5)



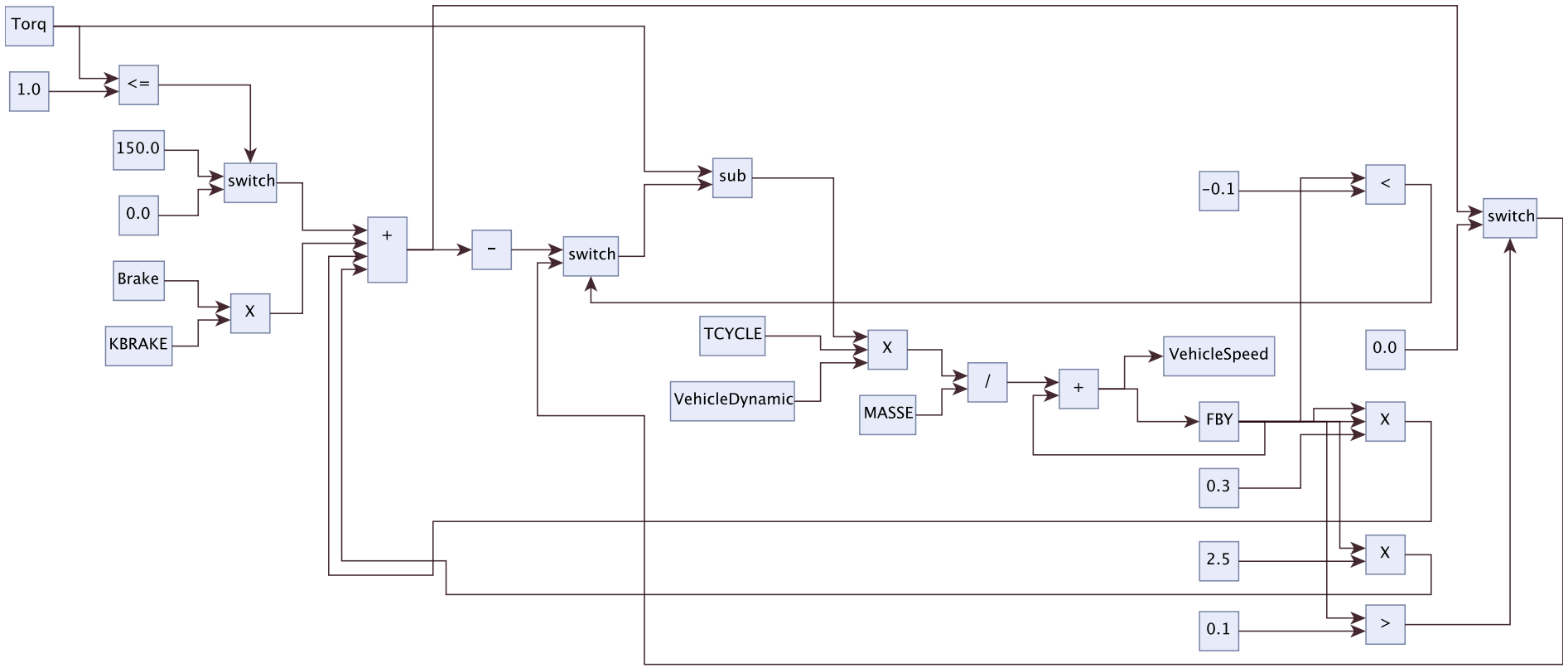
Layout Gallery (4/5): yFiles Layout



Layout Gallery (5/5)

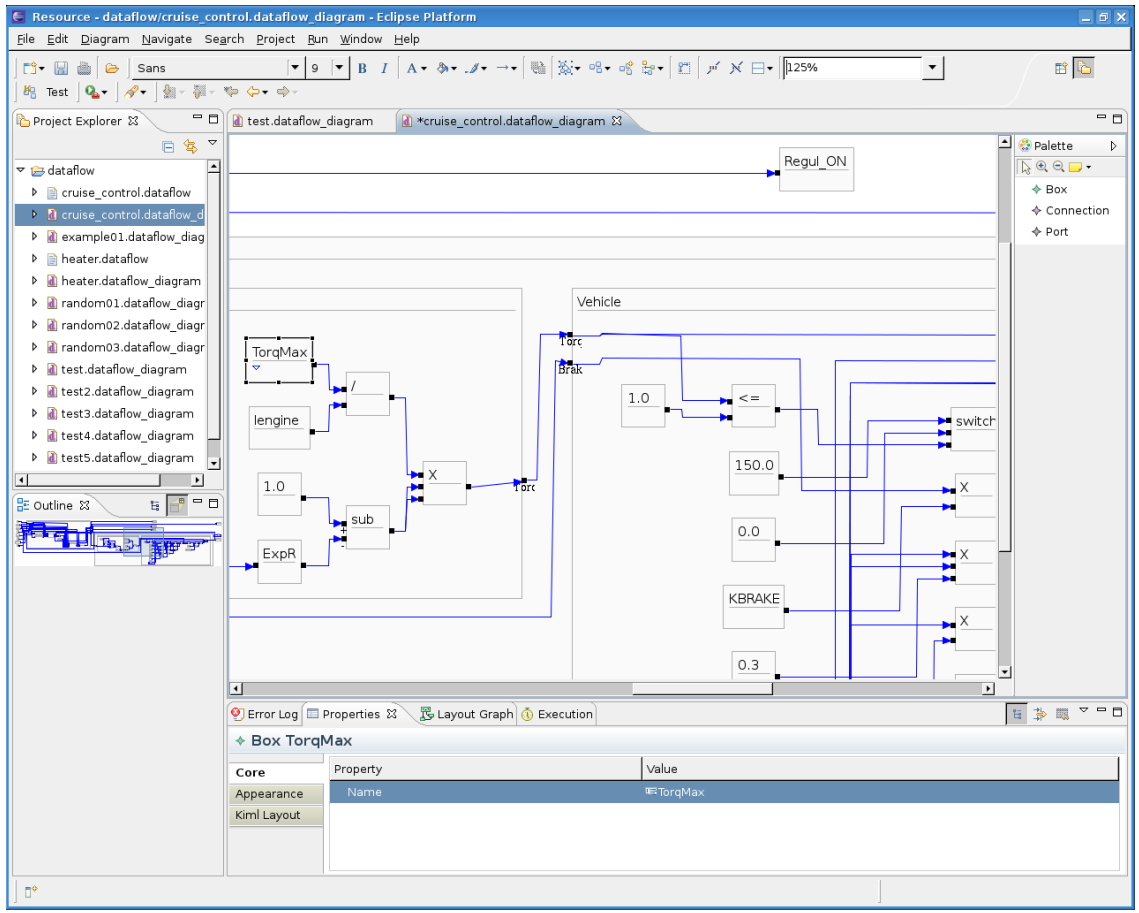


Layout Gallery (5/5): yFiles Layout



Application: KIELER

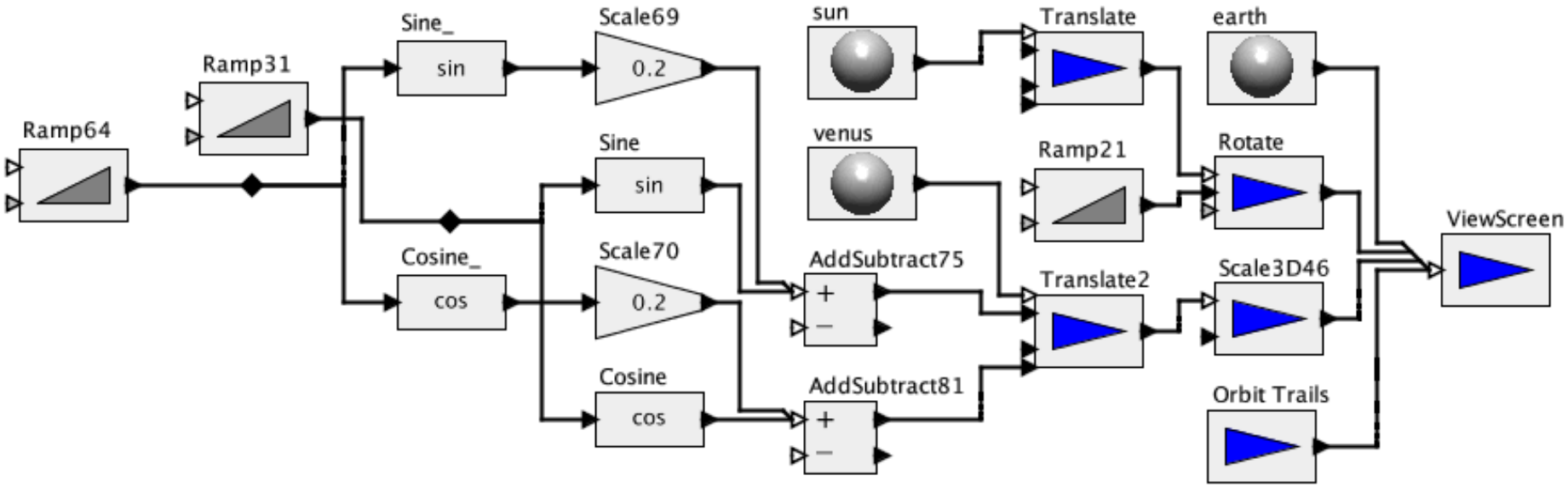
- Research project on graphical modeling, built on Eclipse
- Platform for development of layout algorithms for graphical modeling, open source



www.informatik.uni-kiel.de/rtsys/kieler

Application: Ptolemy

- Research project on concurrent real-time systems (UC Berkeley)



`ptolemy.eecs.berkeley.edu`

Conclusion

- Data flow diagrams impose special requirements on layout algorithms

Conclusion

- Data flow diagrams impose special requirements on layout algorithms
- Main challenges: port constraints, hyperedges, compound graphs

Conclusion

- Data flow diagrams impose special requirements on layout algorithms
- Main challenges: port constraints, hyperedges, compound graphs
- Have presented an extended version of the hierarchical layout approach

Conclusion

- Data flow diagrams impose special requirements on layout algorithms
- Main challenges: port constraints, hyperedges, compound graphs
- Have presented an extended version of the hierarchical layout approach
- Results show suitability for application in graphical modeling
  - Successfully applied in graphical modeling frameworks

Conclusion

- Data flow diagrams impose special requirements on layout algorithms
- Main challenges: port constraints, hyperedges, compound graphs
- Have presented an extended version of the hierarchical layout approach
- Results show suitability for application in graphical modeling
  - Successfully applied in graphical modeling frameworks
- Future work: extend other layout approaches
  - E.g. the topology-shape-metrics approach

Conclusion

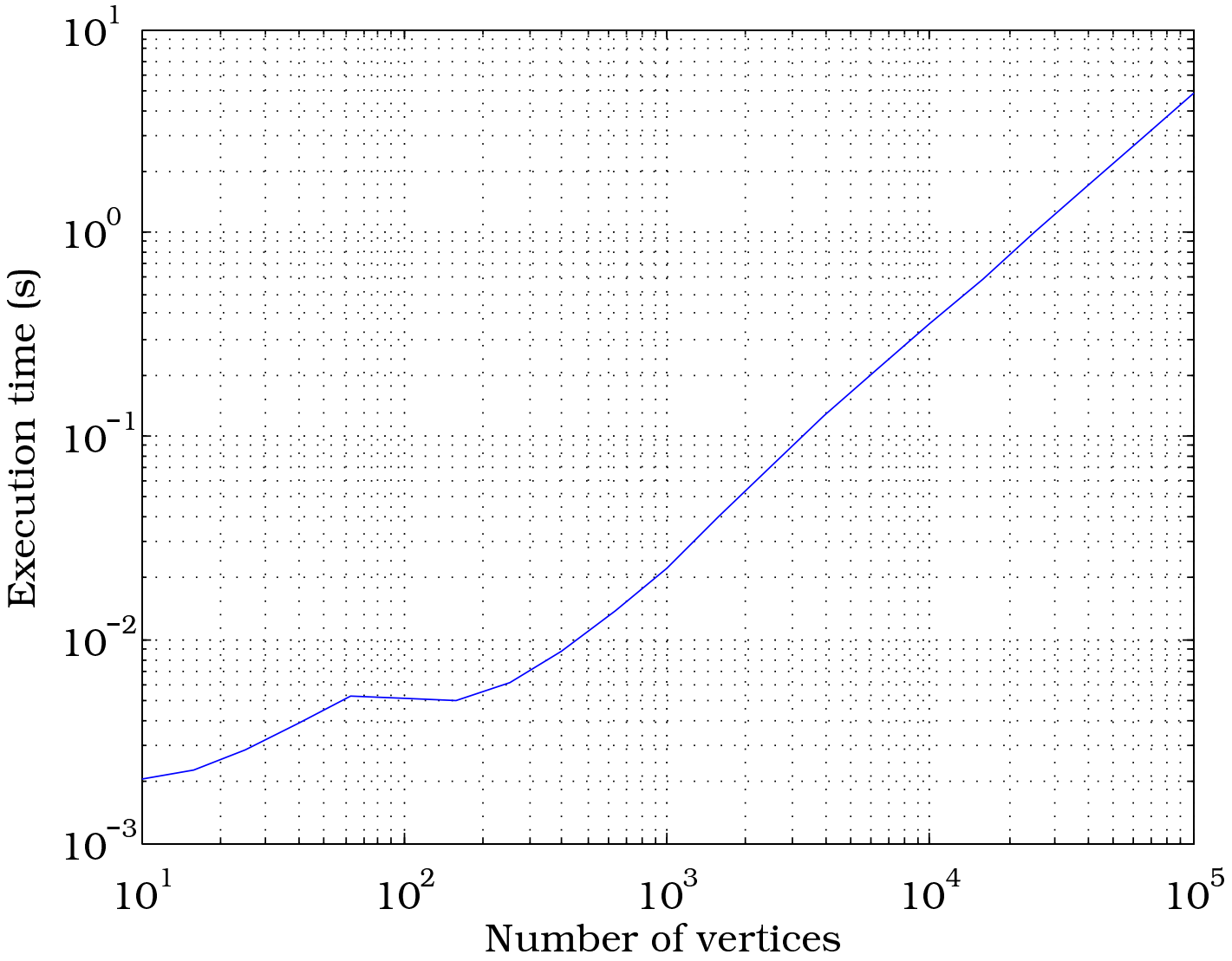
- Data flow diagrams impose special requirements on layout algorithms
- Main challenges: port constraints, hyperedges, compound graphs
- Have presented an extended version of the hierarchical layout approach
- Results show suitability for application in graphical modeling
  - Successfully applied in graphical modeling frameworks
- Future work: extend other layout approaches
  - E.g. the topology-shape-metrics approach

Thanks for your attention

`www.informatik.uni-kiel.de/rtsys/kieler`

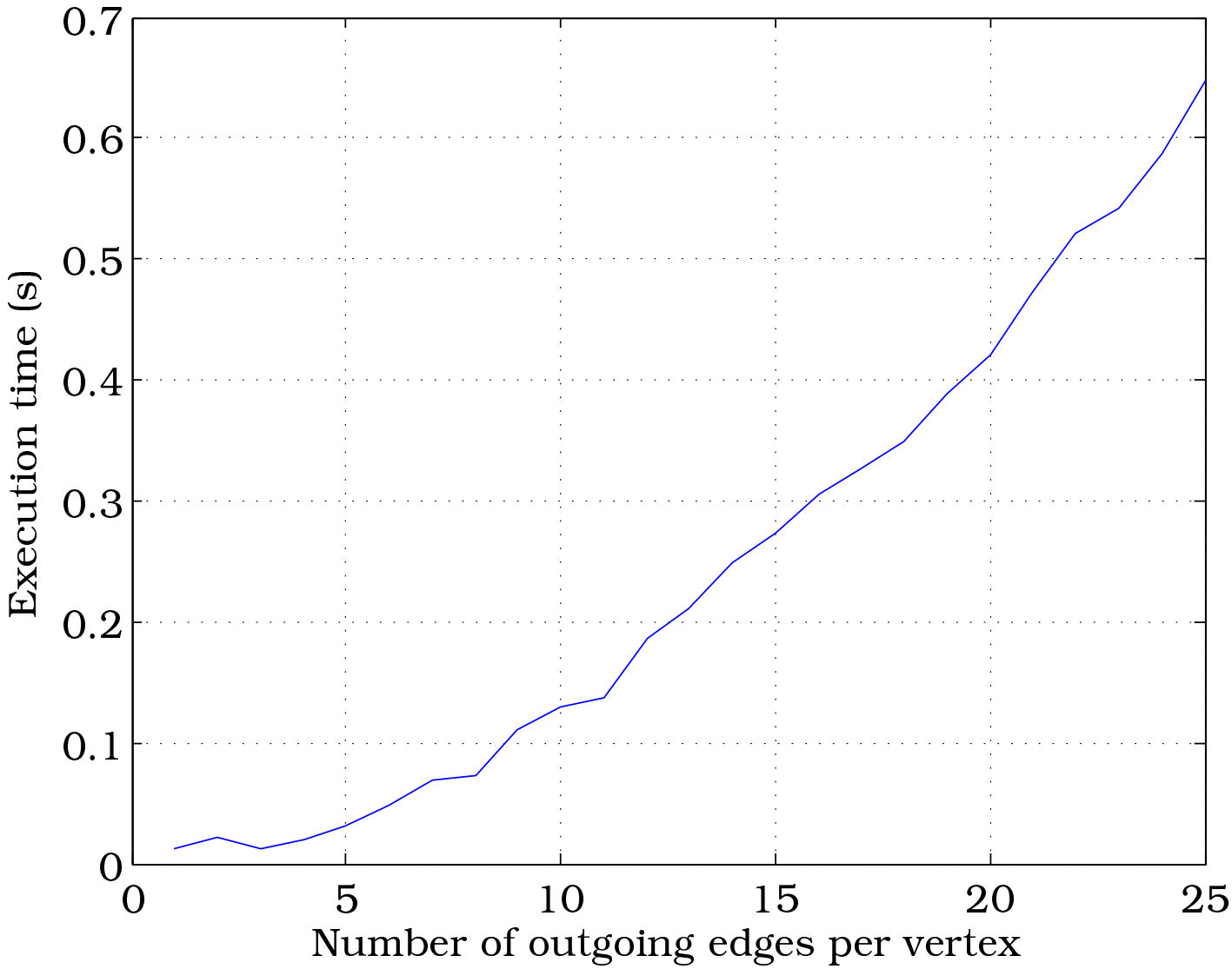
Appendix: Execution Time

Varying number of vertices, one outgoing edge per vertex

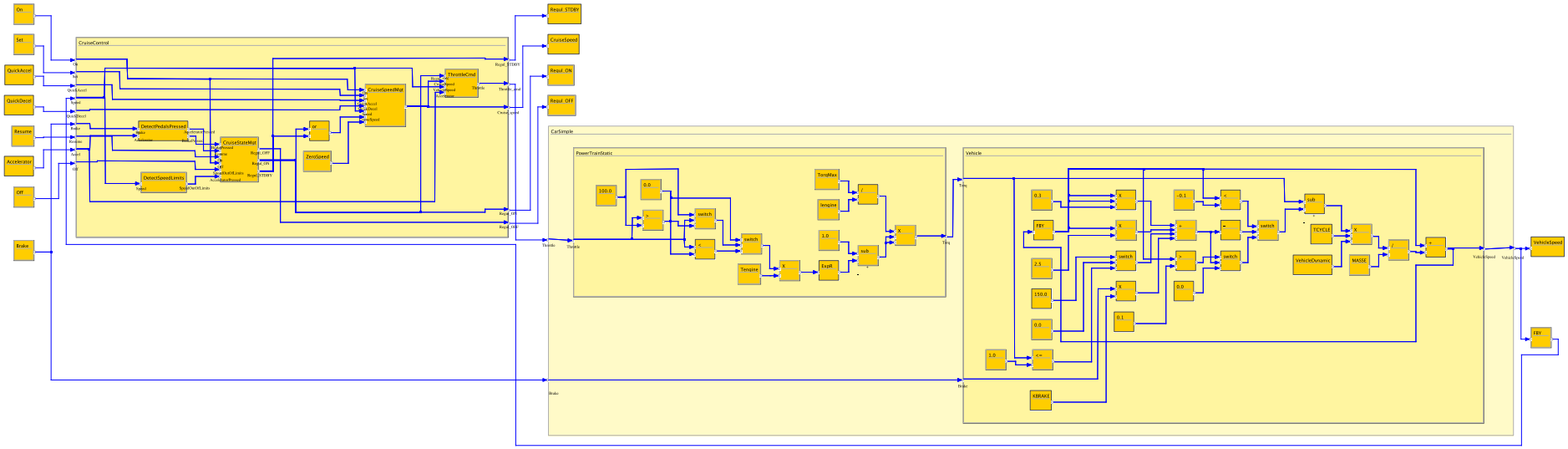


Appendix: Execution Time

100 vertices, varying number of outgoing edges per vertex



Appendix: Compound Diagram Layout



Appendix: Random Diagram Layout