Gladius: An Accurate Method for Fast Path Extraction in Microprocessor Design

Sujeeth Udipi, Sukhdeep Sidhu, Pravin Chandran,
and Walid Elgharbawy (walid.elgharbawy@sun.com *)*
Oracle America, Inc.

Outline

- The Need for Netlist Reduction
- What is Needed?
- Gladius Flow Top-Level Overview
- Gladius Modules:
  - Path Tracer
  - Core Extractor
  - Wire Insertion
  - Side Input Sensitization
  - Create Schematics
- Results and Correlation
- Summary and discussion

The Need for Netlist Reduction

- The ITRS roadmap shows exponential increase in number of transistors over process nodes; almost doubling every 2 years
- Interconnects and parasitics are increasing exponentially as well as adding to the size and complexity of post-layout netlists
- In 40nm and below, a post-layout extraction of a chip can include hundreds of millions to a few billions of elements

Post-layout flat netlist size for small memories can be ~2-3 GB

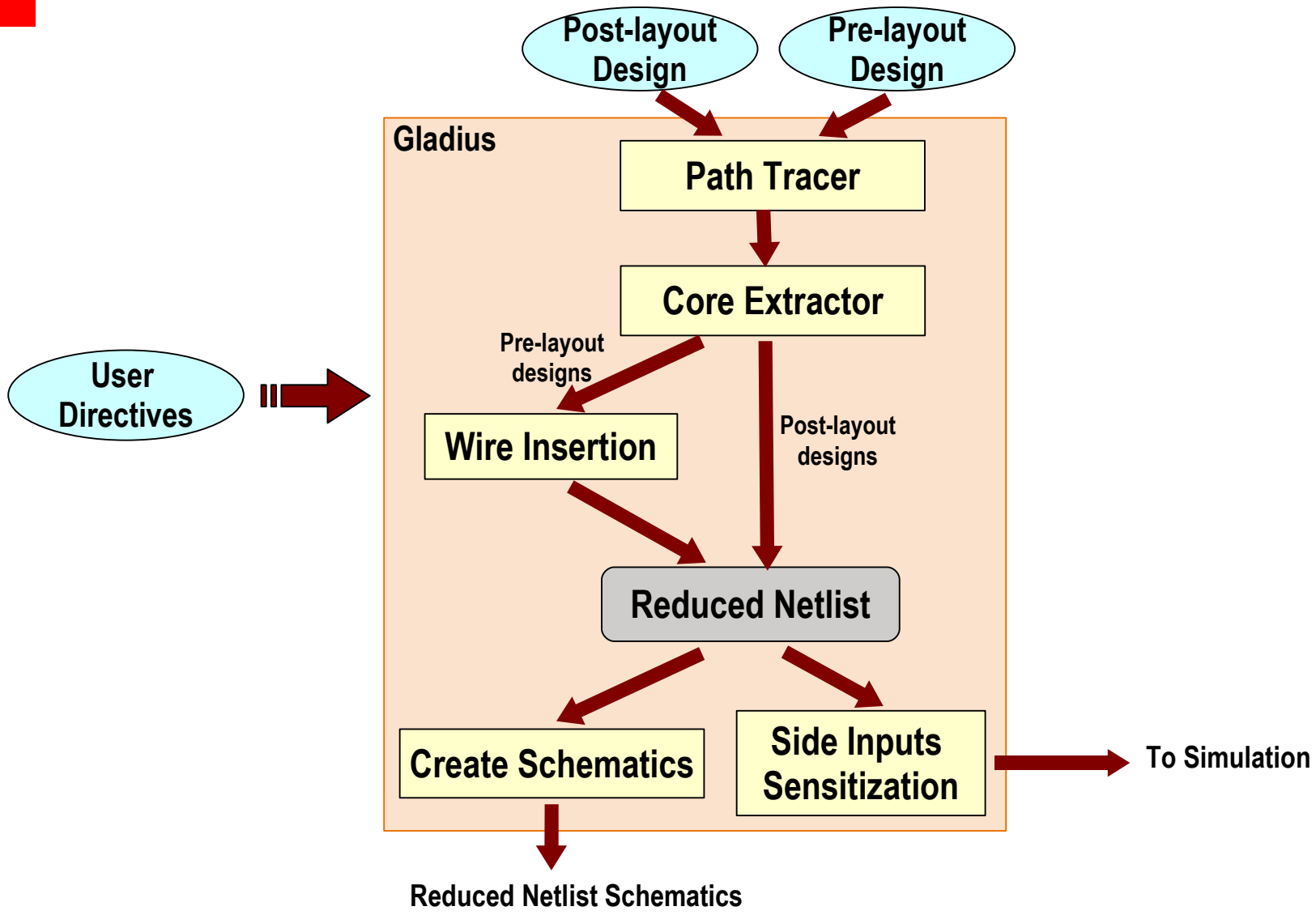
- Large megacell/memory blocks can take many hours to extract and many more hours to simulate

Quick turnaround time for simulation is urgently needed during design cycle!

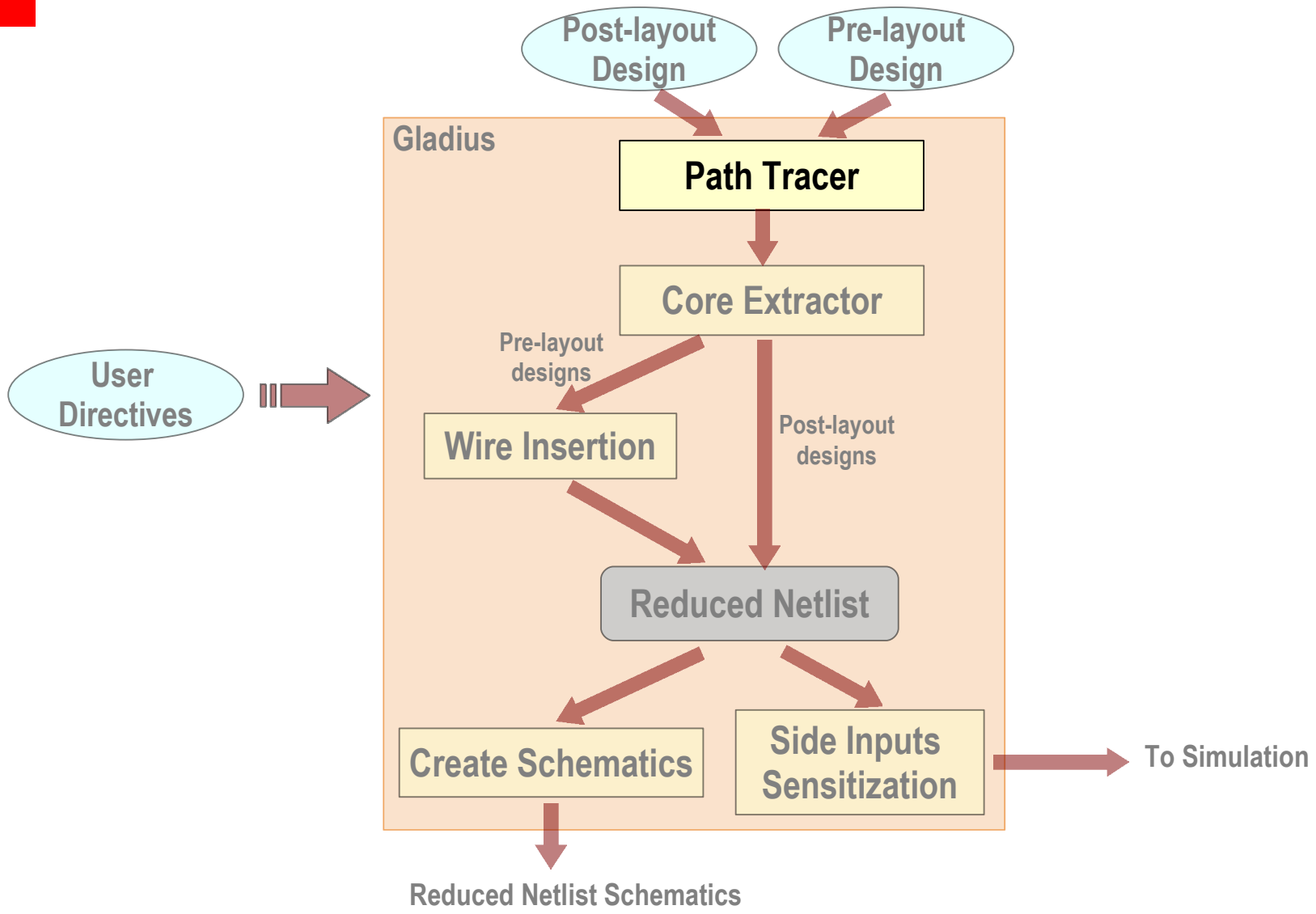
What is Needed?

- A tool that can work on pre-layout netlists as well as post-layout designs
- A tool that can cut out *any* user-specified group of subcircuits
- A lightweight tool, runs in few minutes with a small memory footprint
- Quick turnaround time for enhancements and fixes
- Seamless integration into our in-house CAD tools
- Leverage our in-house CAD tool development capabilities

Gladius Flow Top-Level Overview



Gladius Flow – Path Tracer



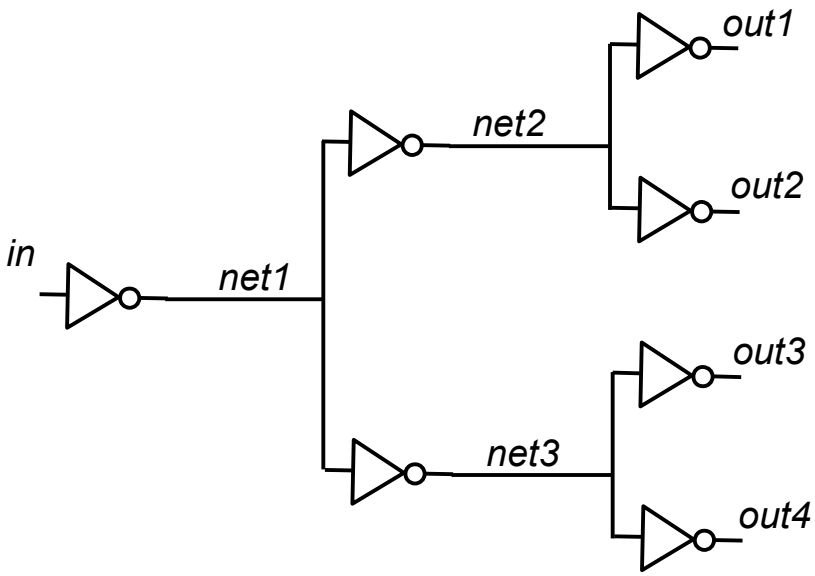
Path Tracer

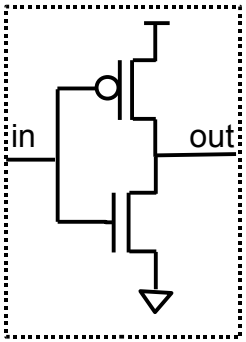
- Generates a list of all the nets and instances to be preserved based on user directives
- To do this Path Tracer utilizes two types of arcs:
  - Cell Arcs: describe all possible traversal arcs through a subcircuit or a cell
  - Net Arcs: arcs between two nets in a design determined through Cell Arcs
- The list of instances is sent to the Core Extractor

Path Tracer Example – Inverter Tree (1)

Basic Approach:

1. Determine “cell_arcs” for all stopcells
2. Determine “net_arcs” for entire design
3. Construct paths, based on “net_arc” info





Cell Arcs:

in → out

Cell arcs are between the pins of a stopcell

Net Arcs:

in → net1
net1 → net2
net1 → net3
net2 → out1
net2 → out2
net2 → out3
net2 → out4

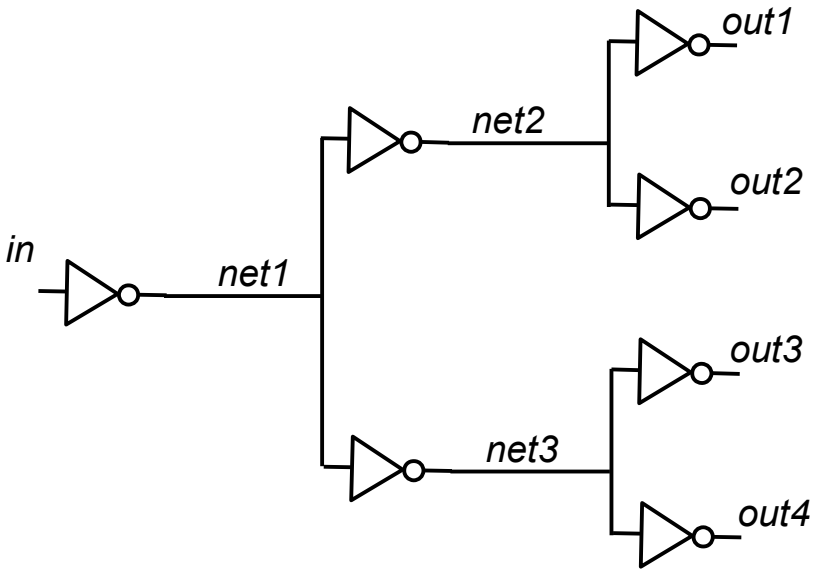
Net arcs are between the nets of the design

All paths from net “in”:

in → net1 → net2 → out1
in → net1 → net2 → out2
in → net1 → net3 → out3
in → net1 → net3 → out4

Path Tracer Example – Inverter Tree (2)

User Directive:
START net: in, END net: out4



Path Traversal:

Initialization:
PATH1=”in”

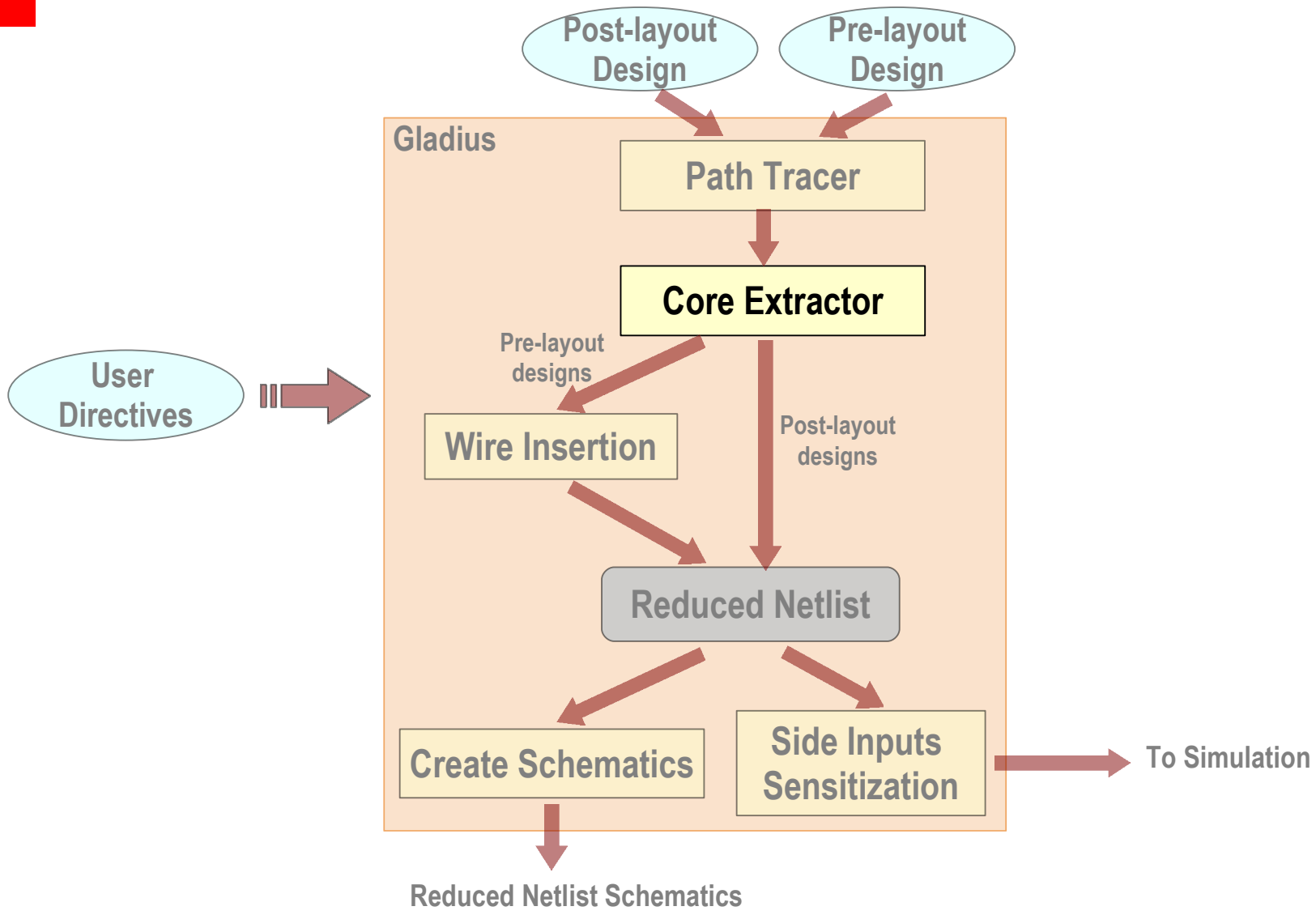
Iteration#1:
PATH1=”in:net1”

Iteration#2:
PATH1=”in:net1:net2”
PATH2=”in:net1:net3”

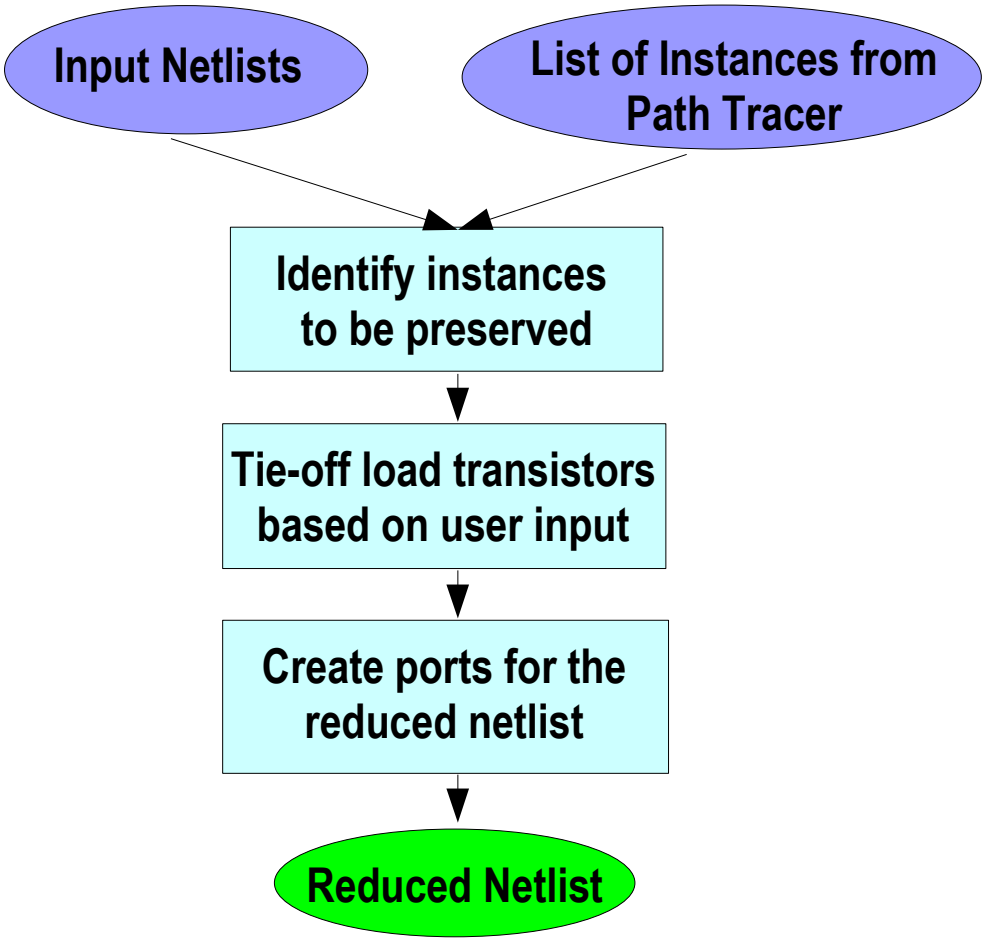
Iteration3:
PATH1=”in:net1:net2:out1”
PATH2=”in:net1:net2:out2”
PATH3=”in:net1:net3:out3”
PATH4=”in:net1:net3:out4”

Path4 is the path to be traced

Gladius Flow – Core Extractor

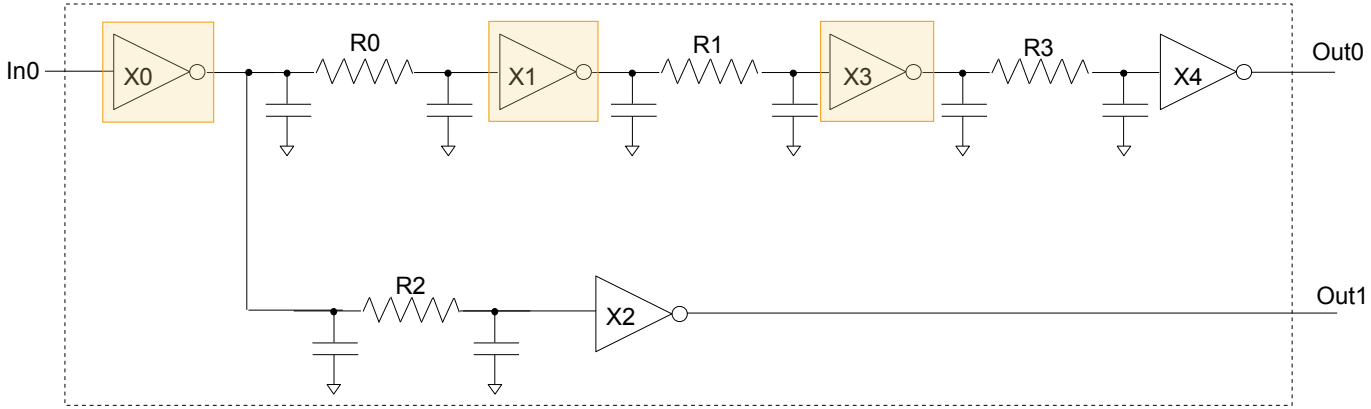


Core Extractor



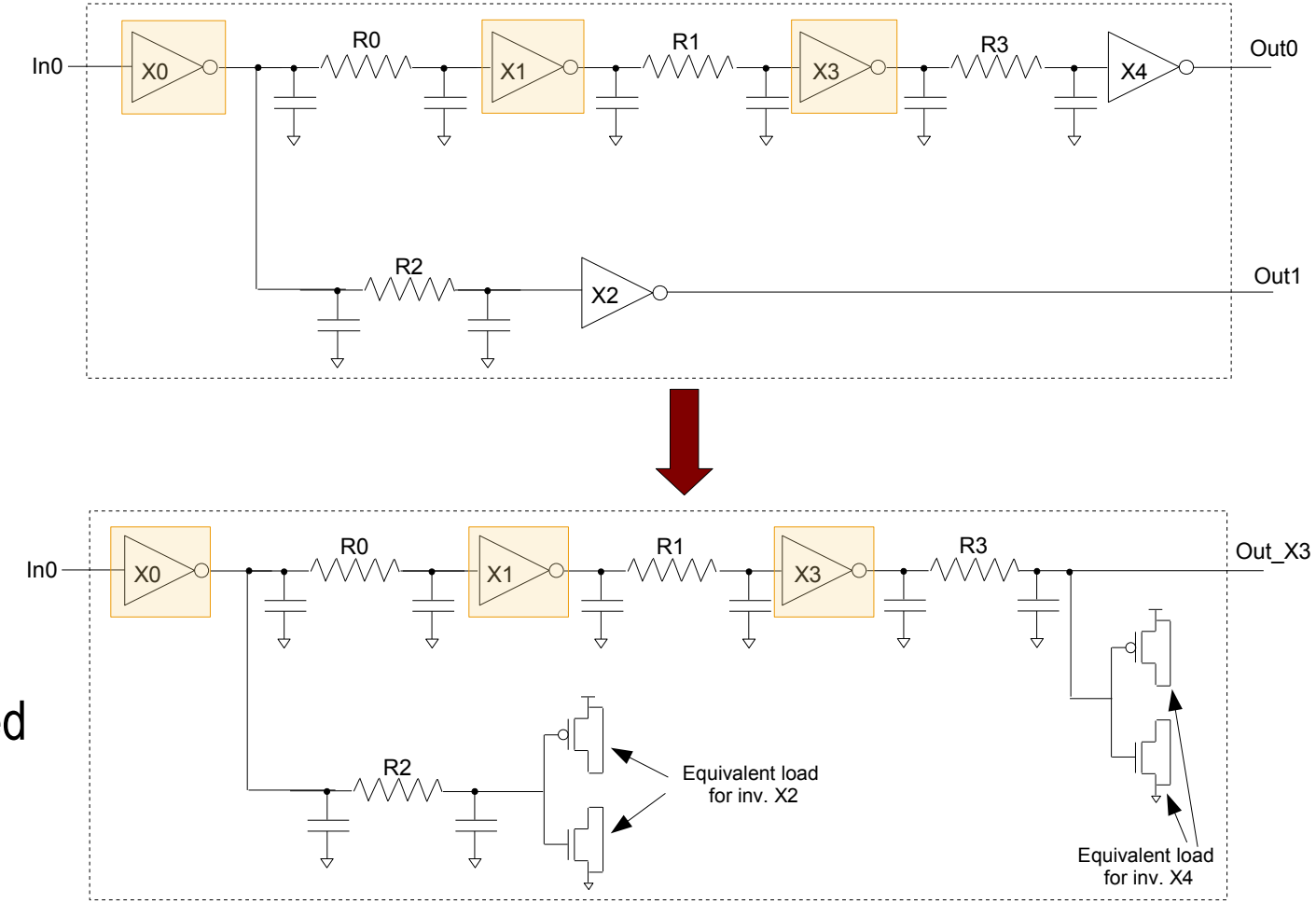
Core Extractor Example

- Input netlist consists of 5 inverters X0-X4
- Required: Extract subckts X0, X1, X3 with load transistors preserved



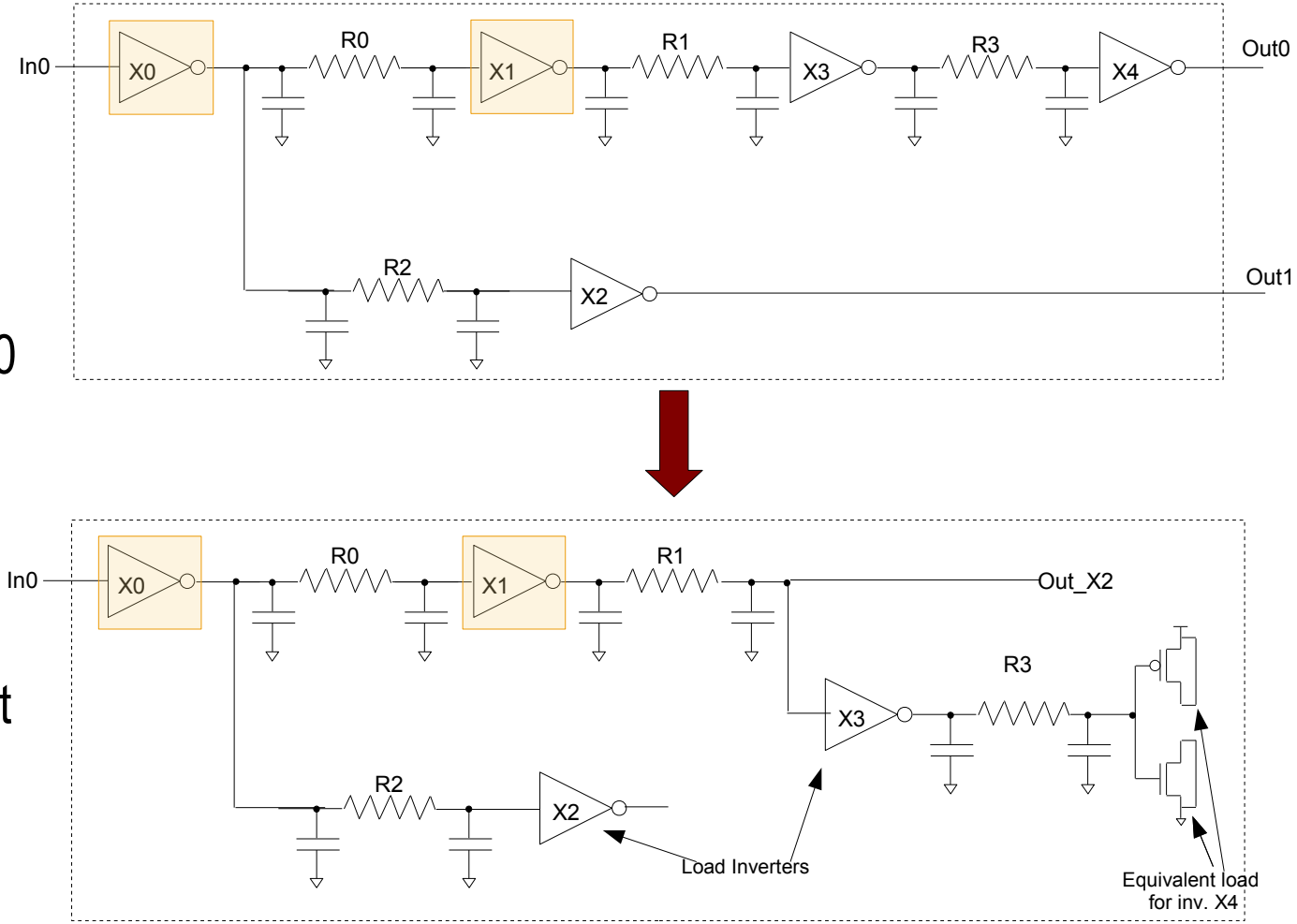
Core Extractor Example – Simple Loading

- Required instances/cells are kept
- Needed input/output pins are created (out_X3)
- Side loads are tied-off

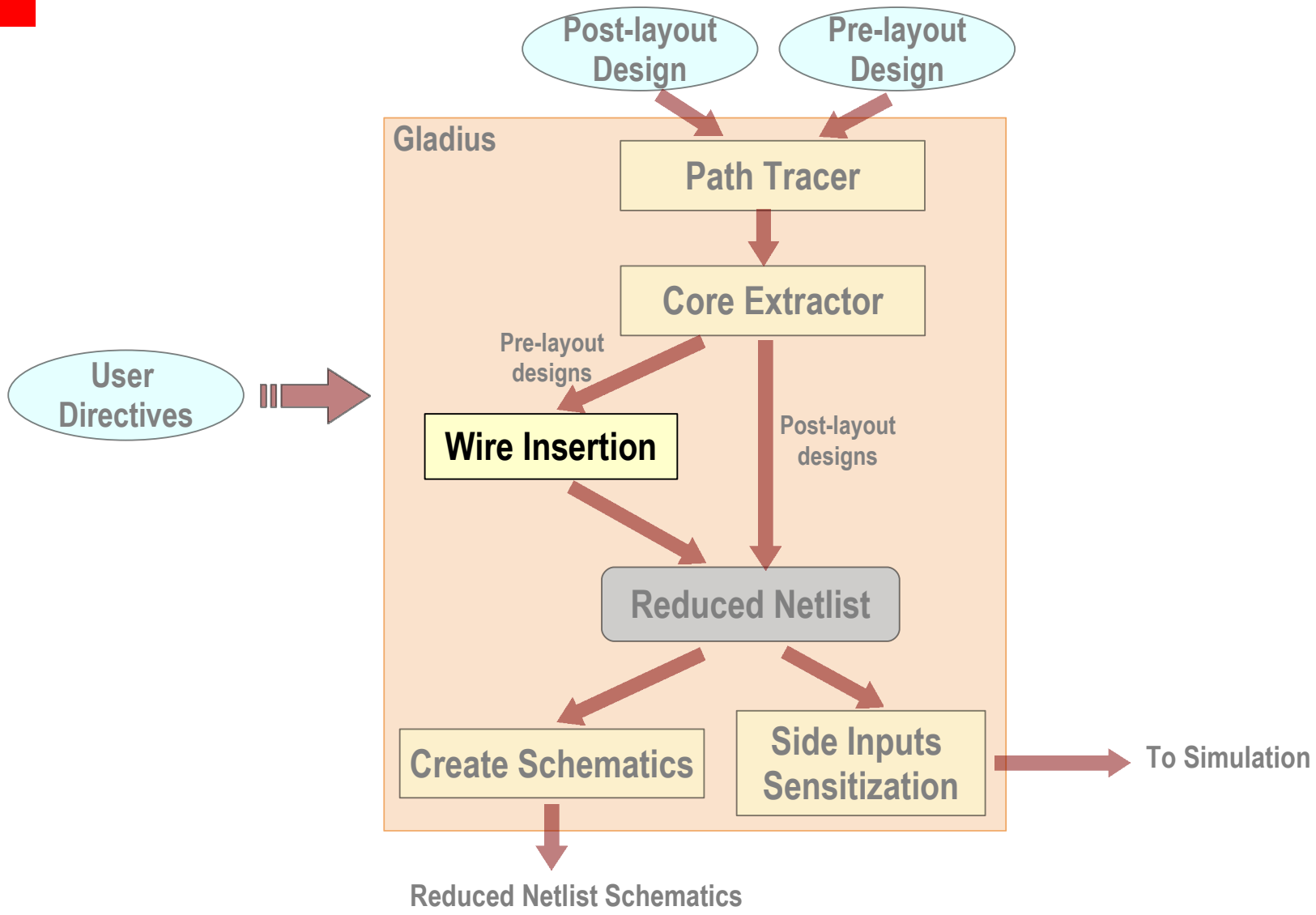


Core Extractor Example - Accurate Loading

- Assume only X0 and X1 are to be kept
- A more accurate side load tie-off is done to account for gate coupling
- Larger netlist size



Gladius Flow – Wire Insertion



Wire Insertion

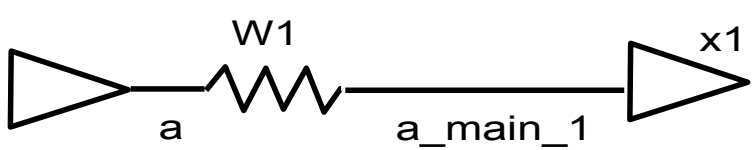
- Adds RC-megawires the reduced netlists in the pre-layout input netlists
- Using user directives, inserts RC-megawires on a given net
- An Example of a directives file to insert a ladder-type wire for net “A”:

```
NET              A
#NO_OF_DRVR      1
#NO_OF_RCVR      3
WIRE_MODEL       daisychain
WIRES            W1,W2,W3
DEFINE_WIRE      W1 M1,l=10u,w=0.5u,sl=0.7u,sr=0.7u
DEFINE_WIRE      W2 M3,l=10u,w=0.5u,sl=0.7u,sr=0.7u,ml=1,mr=1.5,cest=1
DEFINE_WIRE      W3 M2,l=10u,w=0.5u,sl=0.7u,sr=0.7u,ml=1,mr=1.5,cest=1
DRIVER           X0
RECEIVER         X8,X5,X3
END_NET          A
```

Wire Insertion – Configuration Examples (1)

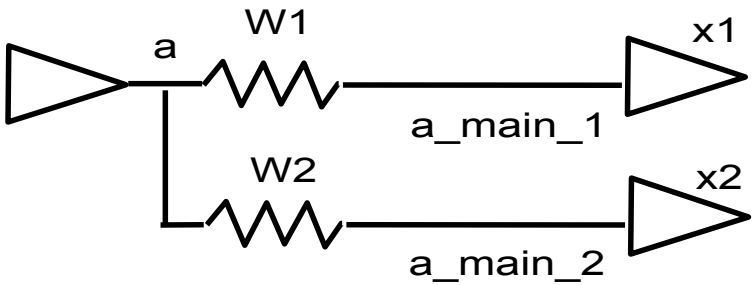
- Can be as simple as a point-to-point (p2p) wire insertion:

```
WIRE_MODEL    p2p
WIRES         W1
RECEIVER      x1
```



p2p with single receiver

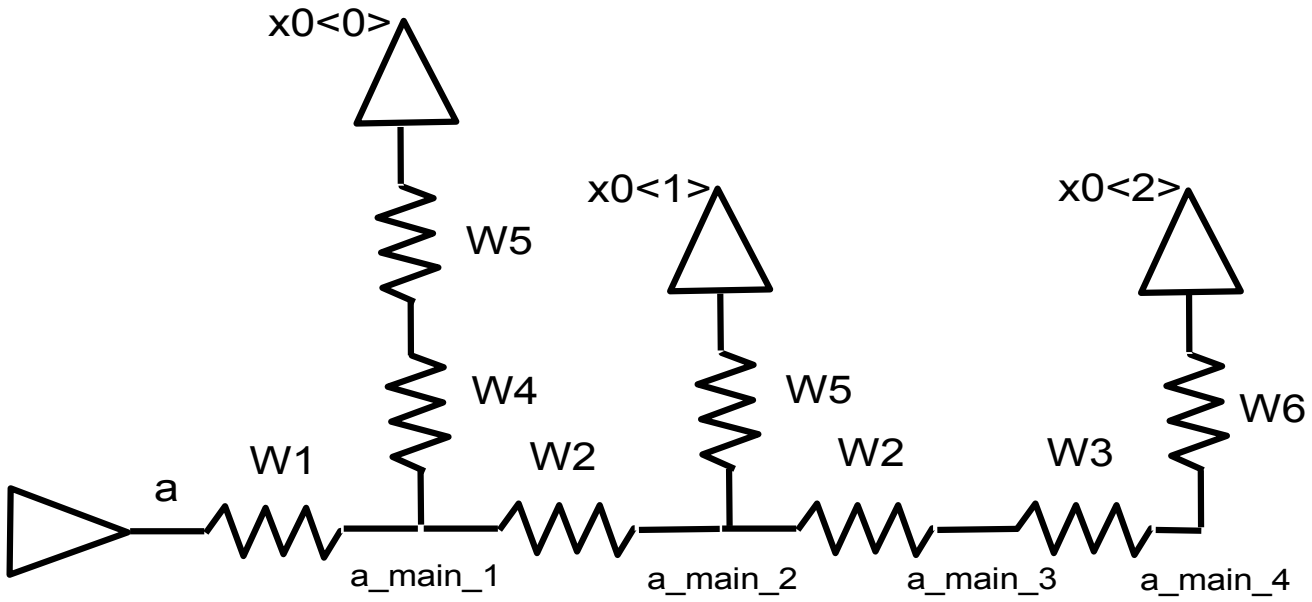
```
WIRE_MODEL    p2p
WIRES         W1,W2
RECEIVER      x1,x2
```



p2p with multiple receiver

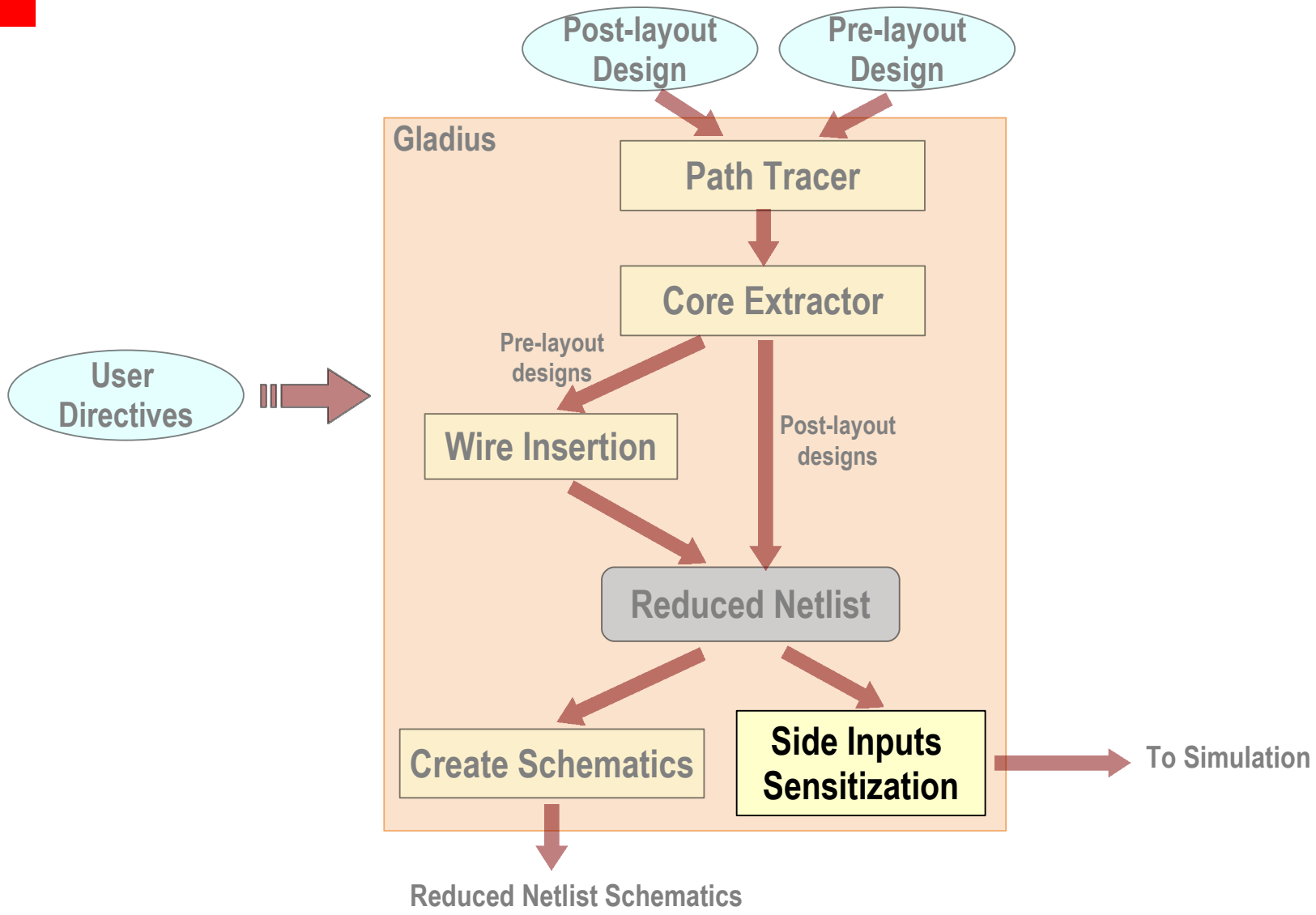
Wire Insertion – Configuration Examples (2)

- Or can be as as complicated as the design requires:



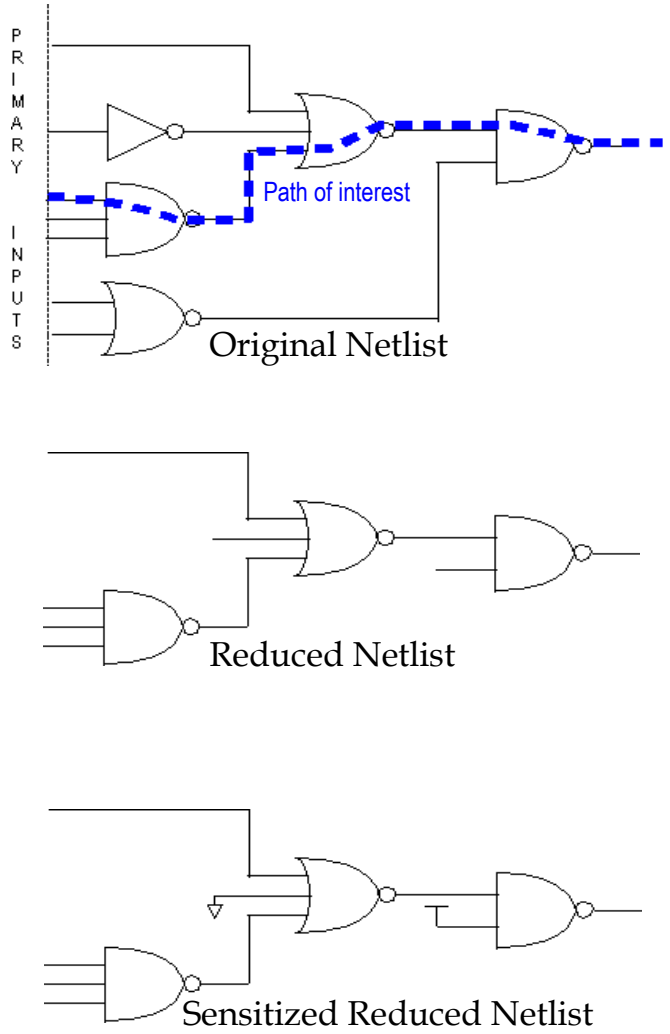

```
WIRE_MODEL    daisychain
WIRES         W1(W4+W5),W2(W5),W2+W3(W6)
RECEIVER      x0<0:2>
```

Gladius Flow – Side Inputs Sensitization

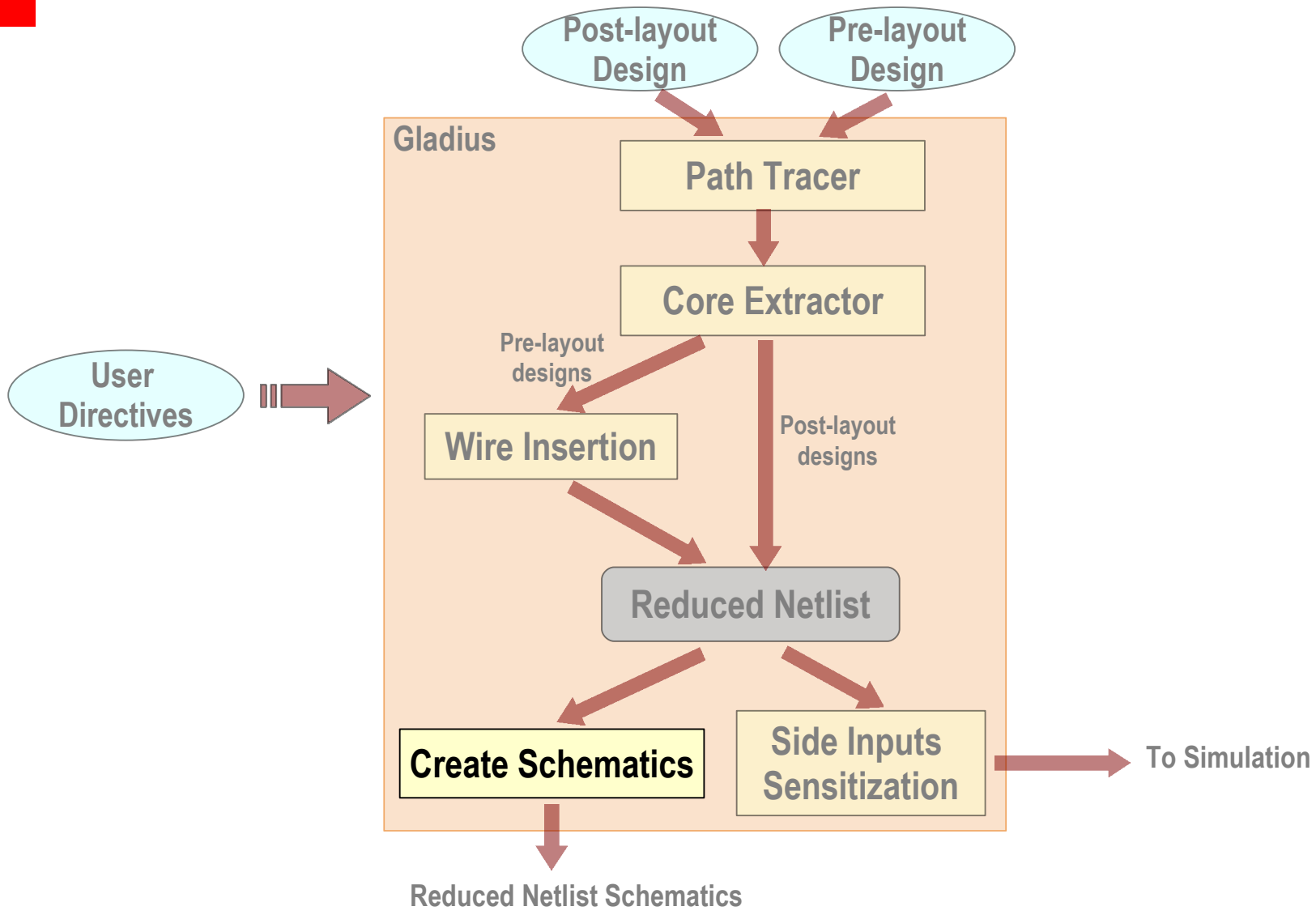


Side Input Sensitization

- Automatically sensitize side inputs to allow logic propagation
- Sensitizations are based on characterization data from libraries
- For custom cells, user directives are used to create sensitizations



Gladius Flow – Create Schematics



Create Schematics

- Generates a verilog netlist corresponding to the reduced design
  - The Cells in this verilog correspond to the stopcells of the Path Tracer
- A submodule (ihdl) converts verilog to schematic
  - ihdl must be able to find the cells in the verilog in the current design library path

Results and Correlation (1)

- In this paper we used 3 SPARC design blocks in 40nm to compare the reduced netlist generated by our tool vs. full block netlist
- Simulations for timing margins were done and individual delay measures were used as a metric for accuracy correlation

Design	Full Netlist			Reduced Netlist		
	Simulation Runtime	Netlist Size	# of Transistors	Simulation Runtime	Netlist Size	# of Transistors
B1	8hr	59MB	20K	18min	11MB	5K
B2	4hr	1.05GB	300K	1hr	174MB	53K
B3	41min	137MB	44.5K	18min	41MB	13K

Results and Correlation (2)

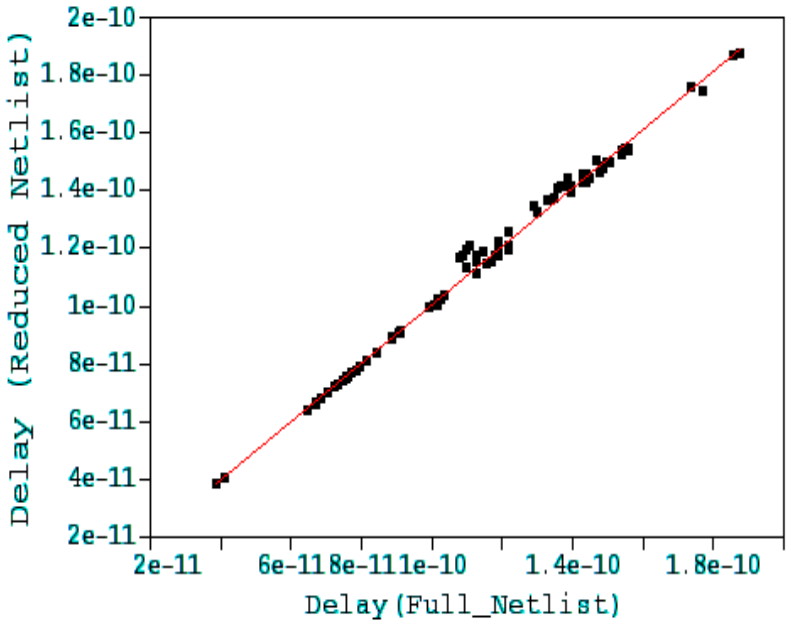
- In this paper we used 3 SPARC design blocks in 40nm to compare the reduced netlist generated by our tool vs. full block netlist
- Simulations for timing margins were done and individual delay measures were used as a metric for accuracy correlation

Design	Full Netlist			Reduced Netlist		
	Simulation Runtime	Netlist Size	# of Transistors	Simulation Runtime	Netlist Size	# of Transistors
B1	8hr	59MB	20K	18min	11MB	5K
B2	4hr	1.05GB	300K	1hr	174MB	53K
B3	41min	137MB	44.5K	18min	41MB	13K

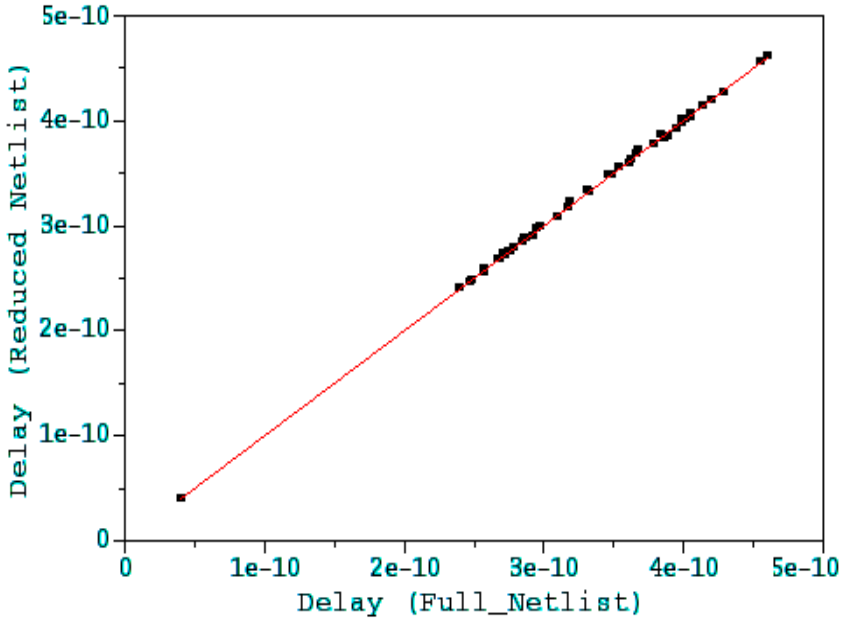
70% - 83% Netlist size reduction and 55% - 96% simulation runtime reduction

Results and Correlation (3)

- Accuracy correlation shows that the tool's output netlist simulation results are always within 1.5% of full netlist



Block 1



Blocks 2&3

Summary

- We developed a tool that allows a designer to selectively cut portions of a design to construct a reduced netlist to be used in different analysis types
- Different modules within the tool are built using proprietary algorithms
- Our tool is not limited to generating netlists for timing paths only; it can cut and generate any subset of the design based on user directives
- Simulations on the reduced netlists produced by the tool are within 1.5% from full netlist simulations with large runtime savings

Acknowledgments

- Pranjal Srivastava
- Shriram Gundala
- Sateesh Medepalli
- Rushang Mehta
- Beena Rangaswamy
- Hemanga Das
- Mohammed Jamil
- Our tool users

Sun

ORACLE

Q & A