Welcome to CircuitOps API Documentation!

`circuitops_api.`**`add_pseudo_fanout_nodes`**`(`*`g`*`, `*`level`*`, `*`num_pins`*`)`

Adds pseudo fan-out nodes at required level

Parameters:
- **g** (*DGL graph object*) – Graph in which nodes are to be added
- **level** (*int*) – Level at which pseudo nodes have to be inserted
- **num_pins** (*int*) – Number of pin nodes in graph

Returns: Graph with pseudo nodes added

Return type: DGL graph object

Example:

graph_mod = add_pseudo_nodes(og_graph, 0, N_pin)

`circuitops_api.`**`calculate_load_cap`**`(`*`self`*`, `*`output_pins`*`, `*`circuit_data`*`)`

Method of **`PinPinEdge`**. Return a PinPinEdge class with output_cap column in its df

Parameters:
- **output_pins** (*List*) – Required. List of pins for which load cap has to be calculated
- **circuit_data** (*Class object of type CircuitData*) – Required. Pass the circuit_data class object

Returns: PinPinEdge class with output_cap column in df

Return type: PinPinEdge class

Example:

cell_arcs_class = pin_pin_edge_class.calculate_load_cap(output_pins, circuit_data)

`circuitops_api.`**`change_graph_bidirectional`**`(`*`g`*`)`

Changes the cell to cell edges as bi-directional

Parameters: **g** (*DGL graph object*) – Graph which has to be changed to bi directional

Returns: bidir_g, graph with bi-directional edges

Return type: DGL graph object

Example:

bidir_graph = change_graph_bidirectional(og_graph)

`circuitops_api.`**`create_singular_graph`**`(`*`g`*`)`

Function to combine two edge type in a graph and form a singular graph.

Parameters: **g** (*DGL graph object.*) – Required. Graph to be merged.

Returns: Singular graph

Return type: DGL graph object

Example:

g_homo = create_singular_graph(g)

`circuitops_api.`**`filter_edge`**`(`*`self`*`, `*`e_type`*`)`

Method of **`CircuitGraph`**. Return a CircuitGraph class with only the mentioned edge type in the graph.

Parameters: **e_type** (*str*) – Required. Pass the edge type to be selected. Can be pin_pin, cell_pin, net_pin, cell_cell or cell_net.

Returns: CircuitGraph class with only given edge type.

Return type: CircuitGraph class

Example:

graph_filtered = circuit_graph.filter_edge(“pin_pin”) `circuitops_api.`**`filter_graph`**`(`*`self`*`, `*`v_mask`*`, `*`e_mask`*`)`

Method of **`CircuitGraph`**. Return a CircuitGraph class after filtering vertices and edge according to input masks.

Parameters:
- **v_mask** (*Array*) – Required. Mask with 1 in only vertices to be selected.
- **e_mask** (*Array*) – Required. Mask with 1 in only edges to be selected.

Returns: CircuitGraph class with only selected vertices and edges.
Return type: CircuitGraph class

Example:

graph_filtered = circuit_graph.filter_graph(v_mask, e_mask)

`circuitops_api.`**`get_arcs`**`(`*`self`*`, `*`arc_type='net'`*`)`

Method of **`PinPinEdge`**. Return a pin to pin edge class after filtering the arc_type

Parameters: **arc_type** (*str*) – Required. Can be either cell or net
Returns: PinPinEdge class with filtered pin to pin df
Return type: PinPinEdge class

Example:

cell_arcs_class = pin_pin_edge_class.filter_pin_arcs(“cell”)

`circuitops_api.`**`get_connected_components`**`(`*`g`*`, `*`threshold=0`*`)`

Returns the connected components in a graph with node count above the threshold.

Parameters:
- **g** (*DGL graph object*) – Graph in which nodes are to be added
- **threshold** (*int*) – Min number of nodes to be in a connected component

Returns: List of connected component graphs
Return type: List

Example:

sub_gs = get_connected_components(og_graph, 100)

`circuitops_api.`**`get_die_boundaries`**`(`*`def_path: str`*`)`

A quick parser to extract die boundaries from DEF.

Parameters: **def_path** – Requied. Pass the path of the DEF file to be parsed.
Returns: llx, lly, urx, ury coordinates.
Return type: Float

Example:

llx, lly, urx, ury = get_die_boundaries(“/path/to/design.def”)

`circuitops_api.`**`get_input_pins`**`(`*`self`*`)`

Method of **`PinProperties`**. Return a list of input pins from a pin properties df.

Returns: List of input pins
Return type: List

Example:

input_pins = pin_props_class.get_input_pins()

`circuitops_api.`**`get_large_components`**`(`*`hist`*`, `*`th=2000`*`)`

Returns the labels for connected components that is larger than threshold

Parameters:
- **hist** (*List*) – Required. Historgram list from label_components function
- **th** (*int*) – Optional. Threshold of component size. Default: 2000

Returns: List of labels of large components
Return type: List

Examples:

comp, hist = label_components(graph, directed=False) labels = get_large_components(hist, 500)

`circuitops_api.`**`get_large_connected_components`**`(`*`self`*`, `*`th=0`*`)`

Method of **`CircuitGraph`**. Return a CircuitGraph class after removing connected components with number of vertices less than threshold.

Parameters:	**th** (*int*) – Min number of vertices to be present in a connected component.
Returns:	CircuitGraph class with only components with size above threshold.
Return type:	CircuitGraph class

Example:

graph_filtered = circuit_graph.get_large_connected_components(100)

`circuitops_api.`**`get_libcellname_edge`**`(`*`self`*`, `*`circuit_data`*`)`

Method of **`PinPinEdge`**. Return a pin pin edge class with libcell name column.

Parameters:	**circuit_data** – Required. Pass the circuit_data class object
Returns:	PinPinEdge class with libcell_name column added.
Return type:	PinPinEdge class

Example:

pin_pin_edge_class_new = pin_pin_edge_class.get_libcellname(circuit_data)

`circuitops_api.`**`get_libcellname_pin`**`(`*`self`*`, `*`circuit_data`*`)`

Method of **`PinProperties`**. Return a pin properties class with libcell name column.

Parameters:	**circuit_data** – Required. Pass the circuit_data class object
Returns:	PinProperties class with libcell_name column added.
Return type:	PinProperties class

Example:

pin_props_class_new = pin_props_class.get_libcellname(circuit_data)

`circuitops_api.`**`get_output_pins`**`(`*`self`*`)`

Method of **`PinProperties`**. Return a list of output pins from a pin properties df.

Returns:	List of output pins
Return type:	List

Example:

output_pins = pin_props_class.get_output_pins()

`circuitops_api.`**`get_port_nets`**`(`*`netlist_path: str`*`)`

A quick parser to extract all I/O netnames from verilog netlist.

Parameters:	**netlist_path** (*str*) – Requied. Pass the path of the netlist file to be parsed.
Returns:	List of port net names.
Return type:	List

Example:

IOnets = get_port_nets("/path/to/netlist")

`circuitops_api.`**`merge_graphs`**`(`*`self`*`, `*`graph_list`*`)`

Method of **`CircuitGraph`**. Merge multiple graph_tool.Graph objects into a single graph.

Parameters:	**graph_list** (*list[graph_tool.Graph]*) – List of graphs to merge.
Returns:	A single merged graph containing all the vertices and edges from input graphs.

Return type: CircuitGraph class

`circuitops_api.`**`merge_tran_cell`**`(`*`self`*`, `*`circuit_data`*`)`

Method of **`PinPinEdge`**. Return a PinPinEdge class after merging tran and cell type columns to its df

Parameters:	**circuit_data** (*Class object of type CircuitData*) – Required. Pass the circuit_data class object
Returns:	PinPinEdge class with pin_tran, cell_type, cell_type_coded columns in df
Return type:	PinPinEdge class

Example:

mereged_class = pin_pin_edge_class.merge_tran_cell(circuit_data)

`circuitops_api.`**`remove_isolated_pins`**`(`*`self`*`, `*`circuit_data`*`)`

Method of **`PinProperties`**. Return a properties class with isolated pins removed from its df.

Parameters:	**circuit_data** – Required. Pass the circuit_data class object
Returns:	PinProperties cass with isolated pins removed.
Return type:	PinProperties class

Example:

pin_props_class_new = pin_props_class.remove_isolated_pins(circuit_data)