PartClusManager
User Guide and Documentation

Authors (Alphabetic order):

Marcelo Danigno (*marcelo@furg.br*)
Mateus Fogaça (*mpfogaca@inf.ufrgs.br*)
Isadora Oliveira (*isoliveira@inf.ufrgs.br*)

Advisors:

Paulo Butzen (*paulo.butzen@ufrgs.br*)
Ricardo Reis (*reis@inf.ufrgs.br*)

List of commands:

```
partition_netlist           -tool name
                            -target_partitions value
                            [-graph_model name]
                            [-clique_threshold value]
                            [-weight_model name]
                            [-max_edge_weight value]
                            [-vertex_weight_range range]
                            [-num_starts value]
                            [-seeds value]
                            [-balance_constraint value]
                            [-coarsening_ratio value]
                            [-coarsening_vertices value]
                            [-enable_term_prop value]
                            [-cut_hop_ratio value]
                            [-architecture value]
                            [-refinement value]
```

```
[-partition_id value]

[-repartition value]
```

Command description:
Divides the netlist into *N* partitions and returns the *id* of the partitioning solution. The command may be called many times with different parameters. Each time, the command will generate a new solution. One or more partitioning solutions can be evaluated using their *id*, with the `evaluate_solution` command, which also finds the best solution given an objective function.

Parameter description:

`-tool`

Description: Defines the partitioning tool.
Availability: Mandatory.
Type: String.
Values: "chaco", "gpmetis" or "mlpart."
Example:

```
partition_netlist -tool gpmetis
```

`-target_partitions`

Description: Number of target partitions.
Availability: Mandatory.
Type: Integer.
Values: [2, 32768].
Example:

```
partition_netlist -target_partitions 4
```

`-threshold`

Description: Max degree of a net decomposed with the clique net model. If using the clique net model, nets with a degree higher than `threshold` are ignored. In the hybrid net model, nets with a degree higher than `threshold` are decomposed using the star model.
Availability: Chaco and GPMetis using clique and star net models. (Optional)
Type: Integer.
Values: [3, 32768] , Default: 50.
Example:

```
partition_netlist -threshold 64
```

`-max_edge_weight`

Description: The max weight of an edge.
Availability: Chaco and GPMetis. (Optional)
Type: Integer.
Values: [1, 32768] , Default: 100.
Example:

```
partition_netlist -max_edge_weight 50
```

`-num_starts`

Description: Number of solutions generated with different random seeds.
Availability: Chaco, GPMetis and MLPart. (Optional)
Type: Integer.
Values: [1, 32768] , Default: 10.
Example:

```
partition_netlist -num_starts 4
```

`-seeds`

Description: Number of solutions generated with set seeds.
Availability: Chaco, GPMetis and MLPart. (Optional)
Type: Vector of integers.
Values: [0; $2^{32} - 1$].
Example:

```
partition_netlist -seeds {10 50}
```

`-balance_constraint`

Description: Max vertex area percentage difference among partitions. E.g., a 50% difference means one partition can hold up to 25% larger area during a 2-way partition.
Availability: Chaco, GPMetis and MLPart. (Optional)
Type: Integer.
Values: [0, 50] , Default: 3.
Example:

```
partition_netlist -balance_constraint 5
```

`-graph_model`

Description: Hypergraph to graph decomposition approach.
Availability: Chaco and GPMetis. (Optional)
Type: String.
Values: "clique", "star" or "hybrid."
Example:

```
partition_netlist -graph_model
```

hybrid

`-weight_model`

Description: Edge weight scheme for the graph model of the netlist.
Availability: Chaco and GPMetis. (Optional)
Type: Integer.
Values: [1; 7] , Default: 7.
1 - $1/(e-1)$ [1]
2 - $4/(e^2-e)$ [2]
3 - $4/(e^2-\ e\ mod\ 2)$ [3]
4 - $6/(e^2+e)$ [4]
5 - $(2/e)^{3/2}$ [5]
6 - $(2/e)^3$ [6]
7 - $2/e$ [6]
"e" → number of pins in the net

Example:

```
partition_netlist -weight_model 2
```

`-coarsening_ratio`

Description: Minimal acceptable reduction in the number of vertices in the coarsening step.
Availability: Chaco. (Optional)
Type: Float.
Values: [0.5; 1.0] , Default: 0.8.
Example:

```
    partition_netlist -tool chaco
-coarsening_ratio 0.7
```

`-coarsening_vertices`

Description: Maximum number of vertices that the algorithm aims to coarsen a graph to.
Availability: Chaco. (Optional)
Type: Integer.
Values: [0; 32768] , Default: 2500.
Example:

```
    partition_netlist -tool chaco
-target_partitions 4
-coarsening_vertices 3000
```

`-enable_term_prop`

Description: Enables Terminal Propagation, which aims to improve data locality. This adds constraints to the KL algorithm, as seen in the Dunlop and Kernighan

Algorithm. Improves the number of edge cuts and terminals with a minimal hit on run-time.
Availability: Chaco. (Optional)
Type: Bool.
Values: [0; 1] , Default: 1.
Example:

```
    partition_netlist -tool chaco
-target_partitions 8 -enable_term_prop 0
```

`-cut_hop_ratio`

Description: Controls the relative importance of generating a new cut edge versus increasing the interprocessor distance associated with an existing cut edge (data locality x cut edges tradeoff).
Availability: Chaco, requires `enable_term_prop` to be 1. (Optional)
Type: Float.
Values: [0.5; 1.0] , Default: 1.0.
Example:

```
    partition_netlist -tool chaco
-enable_term_prop 1 -cut_hop_ratio 0.7
```

`-architecture`

Description: Defines the processor topology for Chaco to use when partitioning the graph. The vertices are then assigned to this topology, with interconnection distance in mind (weight of nets between different sets).
Availability: Chaco. (Optional)
Type: Vector of Integers.
Values: [1; 32768].
Example:

```
    partition_netlist -tool chaco
-architecture {1 5}
```

`-refinement`

Description: Defines how many times a KL refinement is run on Chaco. Has a medium performance hit, but can generate better partitioning results.
Availability: Chaco. (Optional)
Type: Integer.
Values: [1; 32768].
Example:

```
    partition_netlist -tool chaco
-refinement 2
```

`-partition_id`

Description: Enables reading from an already

partitioned result when running the tools. This can be used to generate better results or further partitioning.
Availability: Chaco, GPMetis and MLPart. (Optional)
Type: Integer.
Values: [1; 32768].
Example:

```
    partition_netlist -tool chaco
    partition_netlist -tool chaco
-partition_id 0
```

`-repartition`

Description: Enables incremental partitioning, where only part of the graph is considered when running the partitioning tools. The result from partition_netlist is a merge of a previous result and the new partitioning. This parameter requires a partition_id.
Availability: Chaco, requires `partition_id` (Optional)
Type: Integer.
Values: [1; 32768] , Default: 1.

Example:

```
    partition_netlist -tool chaco
    partition_netlist -tool chaco
-partition_id 0 -repartition 1
```

`evaluate_partitioning` `-partition_ids` *values*

`-evaluation_function function`

Command description:
Evaluates the partitioning solution(s) based on a specific objective function. This function is run for each partitioning solution that is supplied in the `partition_ids` parameter and returns the best one depending on the specified objective (i.e., metric). For the `evaluation_function` "hyperedges" (respectively, "terminals"), the best result would be the one with the lowest total number of hyperedge cuts (respectively, lowest total number of terminals).

Parameter description:

`-partition_ids`

Description: Partitioning solution *id*. These are the

	return values from the `partition_netlist` command. **Availability:** Mandatory. **Type:** Vector of Integers. **Values:** [0; $2^{32}-1$] **Example:** `evaluate_partitioning -partition_ids "0 3"`

`-evaluation_function`	**Description:** The objective function that is evaluated for each partitioning solution.
	Availability: Mandatory.
	Type: Function.
	Values: "terminals", "hyperedges", "size", "area", "runtime", or "hops."
	Example:
	`evaluate_partitioning \`
	`-partition_ids {1 2 7} \`
	`-evaluation_function "terminals"`

`write_partitioning_to_db` `-partitioning_id` *value*

Command description:
Writes the partition id of each instance (i.e. the cluster that contains the instance) to the DB as a property called "partitioning_id."

Parameter description:

`-partitioning_id`	**Description:** Partitioning solution *id*. **Availability:** Mandatory. **Type:** Integer. **Values:** [0; $2^{32}-1$] **Example:** `write_partition_to_db -partitioning_id 0`

`cluster_netlist`	`-tool` *value*
	`-coarsening_ratio`
	`-coarsening vertices`
	`-level`

Command description:
Divides the netlist into N clusters and returns the id of the clustering solution. The command may be called many times with different parameters. Each time, the command will generate a new solution.

Parameter description:

`-tool`	**Description:** Defines the partitioning tool. **Availability:** Mandatory. **Type:** String. **Values:** "chaco", "gpmetis", "mlpart" or “louvain”. **Example:** `cluster_netlist -tool gpmetis`
`-coarsening_ratio`	**Description:** Minimal acceptable reduction in the number of vertices in the coarsening step. **Availability:** Chaco. (Optional) **Type:** Float. **Values:** [0.5; 1.0] , Default: 0.8. **Example:** `cluster_netlist -tool chaco -coarsening_ratio 0.7`
`-coarsening_vertices`	**Description:** Maximum number of vertices that the algorithm aims to coarsen a graph to. **Availability:** Chaco. (Optional) **Type:** Integer. **Values:** [0; 32768] , Default: 2500. **Example:** `cluster_netlist -tool chaco -coarsening_vertices 3000`

`-level`	**Description:** Defines which is the level of clustering to return. **Availability:** Mandatory. **Type:** Integer. **Values:** [0; $2^{32} - 1$] **Example:** `cluster_netlist -tool gpmetis -level 2`

`write_clustering_to_db`	`-clustering_id` *value*

Command description:
Writes the clustering id of each instance (i.e. the cluster that contains the instance) to the DB as a property called "clustering_id."

Parameter description:

`-clustering_id`	**Description:** Clustering solution *id*. **Availability:** Mandatory. **Type:** Integer. **Values:** [0; $2^{32} - 1$] **Example:** `Write_clustering_to_db -clustering_id 1`

`report_netlist_partitions`	`-partitioning_id` *value*

Command description:
Reports the number of partitions and the number of vertices in each one.

Parameter description:

`-partitioning_id`

Description: Partitioning solution *id*.
Availability: Mandatory.
Type: Integer.
Values: [0; $2^{32} - 1$]
Example:

```
report_netlist_partitions
-partitioning_id 0
```

References:

[1] T. Lengauer, *Combinatorial Algorithms for Integrated Circuit Layout*, Wiley-Teubner, New York, 1990.

[2] C. J. Alpert and A. B. Kahng, "Geometric Embeddings for Faster and Better Multi-Way Netlist Partitioning", *Proc. DAC*, 1993, pp. 743-748.

[3] S. W. Hadley, B. L. Mark and A. Vannelli, "An efficient eigenvector approach for finding netlist partitions", *IEEE Trans. CAD* 11(7) (1992), pp. 885-892.

[4] D. J. H. Huang and A. B. Kahng, "When Clusters Meet Partitions: New Density-Based Methods for Circuit Decomposition", *Proc. European Design and Test Conf.*, 1995, pp. 60–64.

[5] J. Frankle and R.M. Karp, "Circuit placement and cost bounds by eigenvector decomposition", *Proc. DAC,* 1986, pp. 414-417

[6] R.-S. Tsay and E.S Kuh, "A unified approach to partitioning and placement", IEEE Trans. Circuits Systems 38(5) (1991), pp. 521-533.