Proceedings of the 18th Conference on Computer Science and Intelligence Systems pp. 1119–1123
DOI: 10.15439/2023F592
ISSN 2300-5963 ACSIS, Vol. 35

Path Length-Driven Hypergraph Partitioning: An Integer Programming Approach

Julien Rodriguez[1,2], François Galea[1], François Pellegrini[2], Lilia Zaourar[1]
0000-0003-3583-0859 - 0000-0002-1594-152X - 0000-0003-3983-6289 - 0000-0002-6660-4347
Université Paris-Saclay, CEA, List, F-91120, Palaiseau, France[1],
Université de Bordeaux, LaBRI et INRIA[2]
name.surname@cea.fr[1]; francois.pellegrini@u-bordeaux.fr

***Abstract*—Circuit prototyping on multi-FPGA (Field Programmable Gate Arrays) platforms is a widely used technique in the VLSI (Very-Large-Scale Integration) context. Due to the ever-increasing size of circuits, it is necessary to use partitioning algorithms to place them on multi-FPGA platforms. Existing partitioning algorithms focus on minimizing the cut size but do not consider the critical path length, which can be degraded when mapping long paths to multiple FPGAs. However, recent studies try to consider the degradation of the critical path and the target topology but these works still use cutting minimization algorithms. In this work, we propose a mathematical model as an integer program (IP) based on the Red-Black Hypergraph model that considers the minimization of the critical path degradation and the target topology. We compare our partitioning results with KHMETIS, a min-cut algorithm, and show a better critical path for many circuit instances.**

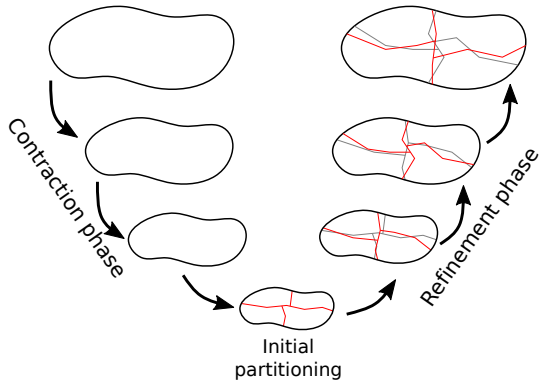

Fig. 1. Multi-level scheme

I. INTRODUCTION

OUR work concerns practical improvements of the electronic circuit design chain. The typical hardware design flow includes different steps, such as floor planning, placement, and routing, that may concern very large logic circuits. To deal with such large circuits, the methods involved may benefit from divide-and-conquer approaches that allow for working locally on separate parts of the circuit, greatly reducing the work on the global circuit. Such a divide-and-conquer approach also enables circuit prototyping on a multi-FPGA platform, where the circuit is too large (in terms of resource consumption) to be implemented on a single FPGA. In such cases, a strong constraint is to mitigate a possible increase in the signal propagation delay of the longest combinatorial path, known as the *critical path*. Indeed, in synchronous circuits, the critical path length determines the maximum frequency at which the circuit may operate; mapping long paths across several FPGAs is likely to degrade the critical path.

Circuit partitioning is both an essential step in the design flow of electronic circuits, and a challenging multi-constraint optimization problem. It must address both the multi-resource issue (i.e., capacity limits on each FPGA and their interconnection links) and the minimization of the critical path degradation.

Traditional partitioning tools use the now classic multi-level scheme (see Fig. 1) consisting of three phases: *coarsening*, *initial partitioning*, and *refinement* [1]. The coarsening phase uses a recursive clustering method to transform the circuit model, a hypergraph, into a smaller one. During the second phase, an initial partitioning is computed on the smallest coarsened hypergraph. Finally, for each coarsening level, the solution for the coarser level is extended to the finer level and then refined using a local refinement algorithm. The initial partitioning algorithm presented in this paper concerns the first step of the multilevel framework described above.

Our work focuses on balanced hypergraph partitioning, in which our objective functions are both *path-cost* minimization and the classical *min-cut* objective that is still relevant to us. The hypergraph model we consider in our research context consists of a union of directed acyclic hypergraphs (DAH) [2]. The global hypergraph is assumed to be connected; otherwise, its disconnected components are processed independently. The source and sink vertices of each DAH (which represent registers and I/O ports) are labeled red, while other vertices are black. Red vertices can be shared by multiple DAHs, which makes the global hypergraph connected. A *path-cost* function models the impact of a cut on the red-to-red paths during partitioning. Each partition of a hypergraph will result in cuts along some paths, inducing additional traversal costs. Our aim is to find a partition of minimum path cost, such that the size of the cut is also minimized. Our research context only considers the paths between two red vertices and a non-uniform cut cost between parts.

The classical approach is to model this problem with a hypergraph, using cost functions that minimize cut size. However, it has been shown in [3] that the cut size does not address the path cost efficiently during the hypergraph partitioning procedure. This is why several authors proposed pre- and/or post-processing steps in order to reduce the degradation of cut paths [3], [4], [5]. In this paper, we devise a dedicated integer programming model that minimizes path cost degradation during partitioning, based on the red-black hypergraph structure, which can be used as an initial partitioning method in a multilevel framework.

The remainder of the paper is organized as follows: Section 2 presents a reminder of our red-black hypergraph structure as well as previous works. Section 3 describes our coarsening scheme before the initial partitioning and the integer programming model. Our experiments are outlined in Section 4. We conclude and give perspectives in Section 5.

II. Preliminaries

In this part, we define the notations and definitions used in this work.

A. Definitions and Notations

Let $\mathcal{H} \stackrel{\text{def}}{=} (\mathcal{V}, \mathcal{A}, \mathcal{W}_v, \mathcal{W}_a)$ be a directed hypergraph, defined by a set of vertices $\mathcal{V}$ and a set of hyperarcs $\mathcal{A}$, with a vertex weight function $\mathcal{W}_v : \mathcal{V} \rightarrow \mathbb{R}^+$ and a hyperarc weight function $\mathcal{W}_a : \mathcal{A} \rightarrow \mathbb{R}^+$. Every hyperarc $a \in \mathcal{A}$ is a subset of vertex set $\mathcal{V}$: $a \subseteq \mathcal{V}$. Let $s^+(a)$ be the source vertex set of hyperarc a, and $s^-(a)$ its sink (destination) vertex set. We consider here, without loss of generality, that each hyperarc has a single source, so $\forall a, |s^+(a)| = 1$. As hyperarcs connect vertices, let $\Gamma(v)$ be the set of neighbor vertices of vertex v, and $\Gamma^-(v) \subseteq \Gamma(v)$ and $\Gamma^+(v) \subseteq \Gamma(v)$ the sets of its inbound and outbound neighbors, respectively.

In the model we propose, hypergraphs that model circuits are be represented as sets of interconnected DAHs, according to a red-black vertex coloring scheme. Red vertices correspond to I/O (Inputs/Outputs) ports and registers, and black vertices to combinatorial circuit components. Let $\mathcal{V}^R \subset \mathcal{V}$ and $\mathcal{V}^B \subset \mathcal{V}$ be the red and black vertex subsets of $\mathcal{V}$, such that $\mathcal{V}^R \cap \mathcal{V}^B = \emptyset$ and $\mathcal{V}^R \cup \mathcal{V}^B = \mathcal{V}$. A hypergraph or sub-hypergraph $\mathcal{H}$ is a DAH iff its red vertices $v_R \in \mathcal{V}^R$ are either only sources or sinks (*i.e.*, $\Gamma^-(v_R) = \emptyset$ or $\Gamma^+(v_R) = \emptyset$), and no cycle path connects a vertex to itself.

Using this definition, we can represent circuit hypergraphs as *red-black hypergraphs*, *i.e.*, sets of DAHs that share some of their red vertices. Let $\mathbf{H}(\mathbf{V}, \mathbf{A}) \stackrel{\text{def}}{=} \{\mathcal{H}_i, i \in \{1 \dots n\}\}$ be a red-black hypergraph, such that every $\mathcal{H}_i$ is a DAH and an edge-induced sub-hypergraph of $\mathbf{H}$. Consequently, $\mathbf{V} = \bigcup_i \mathcal{V}_i$, $\mathbf{A} = \bigcup_i \mathcal{A}_i$, $\mathbf{V}^R = \bigcup_i \mathcal{V}_i^R$, and $\mathbf{V}^B = \bigcup_i \mathcal{V}_i^B$. Moreover, $\forall i, j$ with $i \neq j$, if $\mathcal{V}_{i,j} = \mathcal{V}_i \cap \mathcal{V}_j \neq \emptyset$, then $\mathcal{H}_i$ and $\mathcal{H}_j$ share source and/or sink vertices, *i.e.*, $\mathcal{V}_{i,j} \subset \mathbf{V}^R$.

In this model, the paths in $\mathbf{H}$ to consider when addressing the objective of minimizing path-cost degradation during partitioning are only the paths interconnecting red vertices, as these red-red paths represent register-to-register paths in combinatorial circuits. Since only red vertices are shared between DAHs in $\mathbf{H}$, red-red paths only exist within a single DAH and can never span across several DAHs.

Let us define $\mathbf{P}$ as the set of red-red paths in $\mathbf{H}$, such that $\mathbf{P} \stackrel{\text{def}}{=} \{p | p \text{ is a path in } \mathcal{H} \in \mathbf{H}\}$. From these paths and a function $d_{\max}(u, v)$, which computes the maximum distance between vertices u and v of some DAH $\mathcal{H}$, we can define the longest path distance for $\mathcal{H}$ as: $d_{\max}(\mathcal{H}) \stackrel{\text{def}}{=} \max(d_{\max}(u, v) | u, v \in \mathcal{H})$ and, by extension, for $\mathbf{H}$, as: $d_{\max}(\mathbf{H}) \stackrel{\text{def}}{=} \max(d_{\max}(\mathcal{H}) | \mathcal{H} \in \mathbf{H})$.

A partition Π of $\mathbf{H}$ is a splitting of $\mathbf{V}$ into vertex subsets π_i, called parts, such that:

(i) all parts π_i, given a capacity bound M, respect the capacity constraint:

$$\sum_{v \in \pi_i} \mathcal{W}_v(v) \leq M$$

(ii) all parts are pairwise disjoint:

$$\forall i \neq j, \pi_i \cap \pi_j = \emptyset$$

(iii) the union of all parts is equal to $\mathbf{V}$:

$$\bigcup_i \pi_i = \mathbf{V}$$

Consequently, in our model, the distance between two vertices u and v may increase during partitioning due to the additional cost of routing paths between two (or more) parts. Let $D_{kk'}$ be the penalty associated with parts k and k' such that if u is in part k and v is in part k', then:

$$d^{\Pi}_{\max}(u, v) \geq d_{\max}(u, v) + D_{kk'} \tag{1}$$

For a given partition Π of $\mathbf{H}$, the *path-cost* is defined by the function: $f_p(\mathbf{H}^{\Pi}) = \max(d_{\max}(\mathcal{H}^{\Pi}) | \mathcal{H} \in \mathbf{H})$.

Let a red-black hypergraph $\mathbf{H}$ and a partition Π, the connectivity $\lambda_{\Pi}(a)$ of some hyperarc $a \in \mathcal{A}$ is the number of parts connected by a. If $\lambda_{\Pi}(a) > 1$, then a is said to be cut; otherwise, it is entirely contained within a single part and is not cut. The cut of partition Π is the set $\omega(\Pi)$ of cut hyperarcs, *i.e.*, $\omega(\Pi) \stackrel{\text{def}}{=} \{a \in \mathcal{A}, \lambda_{\Pi}(a) > 1\}$. The cut size is defined as $f_c \stackrel{\text{def}}{=} \sum_{a \in \omega(\Pi)} \mathcal{W}_a(a)$. If all hyperarcs have the same weight (equal to 1), the cut size is equal to $|\omega(\Pi)|$. Another cut metric used by some partitioning tools to measure the quality of partitioning is called *connectivity-minus-one* [6]. The connectivity-minus-one cost function f_λ of some partitioned hypergraph $\mathbf{H}^{\Pi}$ is defined as: $f_\lambda = \sum_{a \in \mathbf{A}} (\lambda_{\Pi}(a) - 1) \times \mathcal{W}_a(a)$.

B. Previous work

Several approaches in the literature have been attempted to improve the performance of circuit partitioning. We present some recent work on circuit partitioning for rapid prototyping that considers performance constraints. Many of these works

attempt to tweak existing *min-cut* partitioning tools, which are used as black boxes, to consider additional constraints. For example, [3] presents a multi-objective approach based on hMETIS. The authors compute the K most critical paths at each partitioning step, using a metric cost that considers the critical path length, the cut number along critical paths, and the weight of the hyperarcs associated with the critical paths. Reference [4] compares a classical method using hMETIS for partitioning followed by a placement algorithm with a derived approach consisting of placement and routing during the partitioning step. The results show better critical path values compared to the two-step approach. More recently, [5] performs some pre- and post-processing on the hypergraph to capture the critical path minimization objective within the cut-size metric, using hMETIS as the partitioning tool. Reference [7], presents an IP model to address the hypergraph partitioning problem. The model is not dedicated to mapping and critical path minimization but to minimize the cut cost.

III. Contributions

We now present our core contribution. The first part consists of a coarsening algorithm to reduce the size of the hypergraph. In the second part, we present our IP model used as initial partitioning.

A. Coarsening method

The heavy-edge matching (HEM) approach for graph coarsening presented in [8] is widely used in hypergraph and graph partitioning tools [9], [10] and yields efficient results in many cases. Our coarsening algorithm is based on a heavy-edge matching approach. It consists of reducing the instance's size while minimizing the merged vertices' weight differences as much as possible. The risk in merging vertices is to end up with disproportionate weights of vertices, which may prevent the initial partitioning from exploring different solutions. However, in the context of critical path minimization, it may be interesting to merge all vertices along the critical path into one large vertex. This method is not necessarily interesting when the circuit contains many critical or semi-critical paths. It is, therefore, necessary to find a compromise between creating a large vertex by securing the cut along the critical path and balancing the fusion to allow a more practical exploration search during the initial partitioning phase. The vertex criticality model the value of the longest path traversing the vertex. Our algorithm groups vertices by criticality to favor the grouping of critical paths. Vertices with a smaller weight are selected to favor balanced coarsening.

B. Integer Program

The objective of the IP model is to minimize the degradation of the critical path, so we need to calculate the maximum degradation among all possible degradations. We also need to model the target topology to consider the different delays between each part. Cut minimization tools do not address these two aspects: path length and topology. Cut minimization tools only limit the connections between parts. As this objective is still essential in practice, we add a second objective to our model: minimizing the connectivity minus one. As the paths between two red vertices do not contain cycle, it is possible to see the chain of black vertices in a path as a sequence of operations/tasks i associated with a job l. In our model, we consider scheduling constraints to minimize the impact of partitioning on the critical path. Given a path (job) $p = v_0, v_1, v_2$, the critical time associated with the path equals $\sum_{v \in p} d_v$. If vertices (tasks) belonging to p are placed in different parts, then a time penalty must be added to the total time of p. A summary of the integer model can be found in Table I, the parameters in Table II, and the variables in Table III. Below is the integer program with two objectives 2a for critical path minimization and 2b for connectivity cost minimization:

TABLE I
INDICES AND SET DEFINITIONS

Set	Definition
$\mathbf{V}$	set of vertices
$\mathbf{E}$	set of hyperedges
J	set of jobs
O_l	ordered set of operations of job l, ($i \in O_l$), where O_{l1} and $O_{ln'}$ are the first and the last elements of O_l
i,i'	vertices/operation index (i, $i' \in \mathbf{V}$)
j, j'	hyperedges index ($j, j' \in \mathbf{E}$)
l, l'	job index ($l, l' \in J$)
k	part index

TABLE II
PARAMETERS DEFINITIONS

Parameter	Definition
n	number of vertices
m	number of hyperedges
h_{ij}	1 if vertex i is connected to hyperedge j, 0 otherwise
c_{kr}	capacity of part k for resource r
q_{ir}	quantity of resource r, required by i
d_i	propagation time of vertices (operation) i
$D_{k,k'}$	delay between part k and k'
$\mathcal{W}_v$	vertex weight
$\mathcal{W}_a$	hyperedge weight

$$\min z_{\max} \tag{2a}$$

$$\min \sum_j \mathcal{W}_a^j \left(\sum_k y_{jk} - 1 \right) \tag{2b}$$

$$\text{subject to} : \sum_k x_{ik} = 1, \quad \forall i \tag{2c}$$

$$h_{ij} x_{ik} \leq y_{jk}, \quad \forall i, j, k \tag{2d}$$

$$\sum_i q_{ir} x_{ik} \leq c_{kr}, \quad \forall k, r \tag{2e}$$

$$\sum_{i,i' \in O_l} d_i + x_{ik} x_{i'k'} D_{kk'} \leq z_l, \quad \forall k, k', l \tag{2f}$$

$$z_l \leq z_{\max}, \quad \forall l \tag{2g}$$

$$x_{ik}, y_{jk} \in \{0,1\}, z_l \in \mathbb{N} \quad \forall i, j, k, l \tag{2h}$$

Constraint 2c states that each vertex is mapped onto one part. Constraint 2d guarantees that y_{jk} equals the connectivity cost

TABLE III
VARIABLES DEFINITIONS

Variable	Definition
x_{ik}	1 iff the vertex i is mapped onto part k, 0 otherwise
y_{jk}	1 iff the hyperedge j has a vertex placed on part k
z_l	completion time of job l
$z_{\max}$	maximum completion time of jobs

associated with hyperedge j. The constraint 2e ensures the capacity constraint is respected. The constraints 2f and 2g determine the value of the delay of the job (path) and the maximum delay (*critical path*). The constraint 2h are the non-negativity and integrity conditions on the variable.

There are symmetries in the solution space in hypergraph partitioning for cut size minimization. Indeed, if there are ω hyperedges between parts, ω remains unchanged regardless of the labels of the parts. On the other hand, in our problem, we are trying to minimize the path cost, which is degraded by routing paths between parts that are not always fully connected. There are models for partitioning graphs and hypergraphs with symmetry-breaking constraints [11]. However, these constraints are too restrictive for the solution space associated with path cost. In our problem, the target topology defines a time penalty associated with path routing. As a result, we cannot consider all partitions with the same subset of vertices but different labels, identical, from a routing point of view. An example can be found in Figure 2. Note that some symmetries exist, for example: if we take the partition a, shown in Figure 2. It is possible to create a partition a' by swapping the vertices of π_0 and π_3 and of π_1 and π_2. Future work will involve improving the model to remove these symmetries.

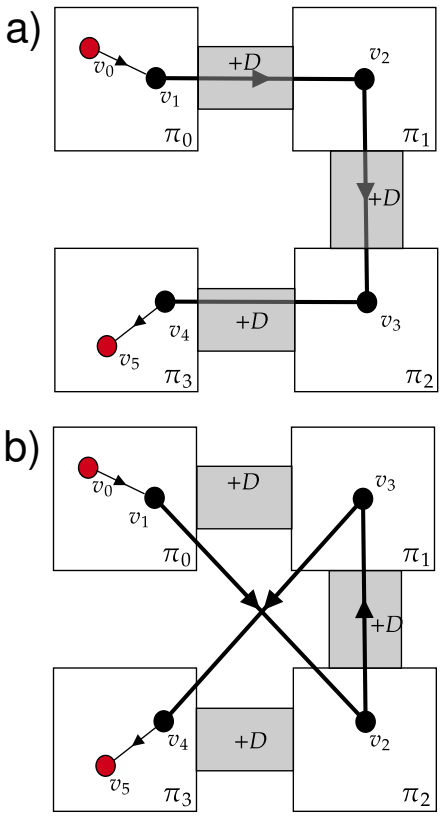

Fig. 2. In this example, the path $p = v_0, v_1, v_2, v_3, v_4, v_5$ is partitioned into 4 parts. In partition a, the path admits a routing penalty of $3D$, where D is the traversal time between parts. In partition b, the routing penalty is $5D$. Since there is no route between π_0 and π_2, we must necessarily pass through π_1 to get there, which gives a cost of $2D$ to get from π_0 to π_2. The same goes for π_1 to π_3. From the point of view of the size of the cut, partition a allows a cost of 3 cut edges, as does partition b. Partitions a and b are identical and symmetrical for cut minimization.

IV. EXPERIMENTAL RESULTS

To validate our models and algorithms, we have performed experiments on benchmarks [12] of logic circuits. These circuits consist of acyclic combinatorial blocks, bounded by their input and output registers. Every combinatorial block can therefore be modeled as a DAH. Their computation time is conditioned by their critical path, defined as the longest path between two registers (*i.e.*, two red vertices). Our work aims at minimizing the degradation of the critical path during partitioning according to the target topology. For each instance, we use topology data to define a traversal cost $d(v)$ for each vertex v, corresponding to the traversal time of a logic element. As the degradation between the parts can be non-homogeneous, we have defined several architecture topologies composed of four elements. The test architecture is a chain $\pi_0, \pi_1, \pi_2, \pi_3$. We did not consider the fully connected topology to highlight the advantage of our topology-aware algorithms over regular partitioners like KHMETIS. To solve the initial partitioning problem, we use Gurobi Optimiser version 9.1.2 with a time limit set to $600s$. During the refinement phase, we use the DKFM [2], a local search algorithm dedicated to minimizing path length. This algorithm is inspired by FM [13], a local search algorithm for minimising the number of hyperedges between two parts. We use KHMETIS rather than HMETIS because HMETIS is based on recursive bipartitioning methods, which often do not respect the balance constraint. We use the maximum criticality as a weight for the hyper-edges as [2] to guide KHMETIS to minimise the number of cuts along the critical path as much as possible.

A. Results

Table IV shows that our approach gives better results. Indeed, the first coarsening step allows the grouping of the most critical vertices while maintaining a balance in the reduced hypergraph. Finally, since the initial partitioning considers the topology, it allows for finding an appropriate placement before the refinement phase. For instances B14 and B17, the time limit is not sufficient for Gurobi to find a good solution. A method needs to be found to better reduce the size of the instance while retaining sufficient criticality information for the integer program. Table V shows us a better performance

TABLE IV
RESULTS FOR PATH-COST f_p IN NANO-SECONDE (NS)

Instance	KHMETIS (ns)	Multilevel+IP+DKFM (ns)
b01	60	**50**
b02	**30**	**30**
b03	50	**40**
b04	**60**	**60**
b05	**50**	**50**
b06	40	**30**
b07	90	**60**
b08	90	**70**
b09	**40**	**40**
b10	**80**	**80**
b11	80	**70**
b12	**40**	**40**
b13	30	**20**
b14	**40**	100
b17	**200**	215.52

of KHMETIS for the function f_λ. Note that our approach sometimes allows a better solution for both f_p and f_λ.

TABLE V
RESULTS FOR CONNECTIVITY f_λ

Instance	KHMETIS	Multilevel+IP+DKFM
b01	**15604**	24288
b03	13903	**10922**
b03	**21297**	29962
b04	**38320**	103659
b05	**30753**	70329
b06	20526	**19043**
b07	**30372**	114329
b08	**28170**	44511
b09	24830	**24445**
b10	**32989**	41600
b11	**49883**	57329
b12	**30743**	99448
b13	**4567**	6000
b14	**214772**	1578740
b17	**846531**	3149961

V. CONCLUSION

In this paper, we present a multilevel approach to the problem of red-black hypergraph (circuit) partitioning on not fully connected topologies. Our approach consists of exploiting the vertices' criticality to group the critical paths in the same part during the coarsening phase. Finally, we propose a mathematical model considering the two objectives: f_p and f_λ for the initial partitioning. For the refinement, we use the DKFM algorithm. Our results show that our approach is better at minimizing f_p than a min-cut partitioning tool, even if it is oriented towards the criticality of hyperarcs. It may be interesting to test our approach on other more extensive benchmarks, as well as to test other coarsening algorithms to improve the results of initial partitioning based on our IP.

REFERENCES

[1] G. Karypis, R. Aggarwal, V. Kumar, and S. Shekhar, "Multilevel hypergraph partitioning: applications in VLSI domain," *IEEE Transactions on VLSI Systems*, vol. 7, no. 1, pp. 69–79, 1999.

[2] J. Rodriguez, F. Galea, F. Pellegrini, and L. Zaourar, "A hypergraph model and associated optimization strategies for path length-driven netlist partitioning," in *International Conference on Computational Science*. Springer, 2023, pp. 652–660.

[3] C. Ababei, S. Navaratnasothie, K. Bazargan, and G. Karypis, "Multi-objective circuit partitioning for cutsize and path-based delay minimization," in *IEEE/ACM ICCAD 2002.*, 2002, pp. 181–185.

[4] M.-H. Chen, Y.-W. Chang, and J.-J. Wang, "Performance-driven simultaneous partitioning and routing for multi-fpga systems," in *2021 58th ACM/IEEE DAC*, 2021.

[5] S.-H. Liou, S. Liu, R. Sun, and H.-M. Chen, *Timing Driven Partition for Multi-FPGA Systems with TDM Awareness*. New York, NY, USA: Ass. Comp. Mach., 2020, p. 111–118. [Online]. Available: https://doi.org/10.1145/3372780.3375558

[6] U. Çatalyürek, K. Devine, M. Faraj, L. Gottesbüren, T. Heuer, H. Meyerhenke, P. Sanders, S. Schlag, C. Schulz, D. Seemaier, and D. Wagner, "More recent advances in (hyper)graph partitioning," *ACM Computing Surveys*, vol. 55, no. 12, mar 2023. [Online]. Available: https://doi.org/10.1145/3571808

[7] D. Kucar, S. Areibi, and A. Vannelli, "Hypergraph partitioning techniques," *DYNAMICS OF CONTINUOUS DISCRETE AND IMPULSIVE SYSTEMS SERIES A*, vol. 11, pp. 339–368, 2004.

[8] G. Karypis and V. Kumar, "Analysis of multilevel graph partitioning," in *Proceedings of the 1995 ACM/IEEE Conference on Supercomputing*, ser. Supercomputing '95. New York, NY, USA: Association for Computing Machinery, 1995, p. 29–es. [Online]. Available: https://doi.org/10.1145/224170.224229

[9] K. George and K. Vipin, "Hmetis: a hypergraph partitioning package," *ACM Transactions on Architecture and Code Optimization*, 1998.

[10] F. Pellegrini, "Scotch and PT-Scotch Graph Partitioning Software: An Overview," in *Combinatorial Scientific Computing*, O. S. Uwe Naumann, Ed. Chapman and Hall/CRC, 2012, pp. 373–406. [Online]. Available: https://hal.inria.fr/hal-00770422

[11] P. Bonami, V. H. Nguyen, M. Klein, and M. Minoux, "On the solution of a graph partitioning problem under capacity constraints," in *Combinatorial Optimization: Second International Symposium, ISCO 2012, Athens, Greece, April 19-21, 2012, Revised Selected Papers 2*. Springer, 2012, pp. 285–296.

[12] F. Corno, M. Reorda, and G. Squillero, "RT-level ITC'99 benchmarks and first ATPG results," *IEEE Design & Test of Computers*, vol. 17, no. 3, pp. 44–53, 2000.

[13] C. Fiduccia and R. Mattheyses, "A linear-time heuristic for improving network partitions," in *19th DAC*, 1982.