ULB ECOLE POLYTECHNIQUE DE BRUXELLES beams

Automated System Partitioning for Efficient 3D Circuit Integration

Research Progress Report

Quentin Delhaye – September 2017

Who does not want a more powerful and compact chip to process their funny cat videos? The industry has been fulfilling this wish for decades, but struggles to keep it financially viable. It is becoming obvious that the classic 2D architectures are being limited by their own technology. We need to go beyond the 2D paradigms and explore the third dimension. Numerous research groups are working on different aspects of this vast and problem-prone subject.

We are currently working on automating the partitioning of 2D architectures that is still mostly done by hand. To do so, we are developing a solution interfacing with popular production sofwtares and standards that will process the placed and routed 2D design, extract the underlying hypergraph and statistics, cluster it, partition it and re-generate netlists for each die.

As of now, we have a hypergraph and statistics extractor as well as a naive clustering solution for bipartitioning any architecture. Although existing partitioning libraries were used, new weights metrics have been studied. The impact of clustering granularity on inter- and intra-cluster connectivity has also been under our scope.

Some intermediate results showed a gain of up to 77% in total wire length on some architectures when going 3D, and clusters containing 1000 to 5000 gates appears to be the sweet spot in terms of inter-cluster nets versus inter-cluster wire length.

Contents

Glossary

clique Sub-graph among which all vertices are connected to each other.

Jaccard index The Jaccard index of two sets A and B can be defined as $\frac{|A \cap B|}{|A \cup B|}$.

k-clique Clique composed of k nodes.

wafer In integrated circuits fabrication, the wafer is the substrate on which the circuits are built.

yield In integrated circuits fabrication, the yields denotes the proportion of properly functioning chips on a wafer.

Acronyms

BEOL back-end-of-line.

CTS clock tree synthesis.

D2D die-to-die.

D2W die-to-wafer.

DEF Design Exchange Format.

EC Edge Coarsening.

F2B face-to-back.

F2F face-to-face.

FEOL front-end-of-line.

FM Fiduccia-Mattheyses.

HEC Hyperedge Coarsening.

IC Integrated Circuit.

ITRS International Technology Roadmap for Semiconductors.

KL Kernighan-Lin.

LEF Library Exchange Format.

M3D monolithic 3D IC.

MHEC Modified Hyperedge Coarsening.

MIV monolithic inter-tier via.

P&R Place and Route.

PDN power delivery network.

SA Simulated Annealing.

TS Tabu Search.

TSV through-silicon-via.

W2W wafer-to-wafer.

WL wire length.

Imagine you are the heir to an antic civil contractors family. Since the dawn of time, your family has managed their terrain as a parking lot. First it was for horse-drawn caravans, bulky and rare. Along the years, the vehicles became less ponderous and cheaper, thus more common. Nowadays, they are as small as they can get and ever cheaper. What are now called "cars" fill your terrain and more are piling up at the entrance. You need them on your domain if you want to prosper. Will you still wait for more clever engineers to develop smaller cars, or will you finally build a new floor and go 3D?

Chapter 1

Motivation: Why do we want to go 3D?

Stability, security, performance, we always want more. The same goes for Integrated Circuits (ICs). They are everywhere: in your dishwasher, your phone, your car, your heating system, maybe even inside your body.

Take the figure 1.1. This little piece of technology is making the world go round. But the world is greedy, it always wants more. For a very long time, more meant smaller transistors, but in the recent years, this scaling faced ever complex challenges.

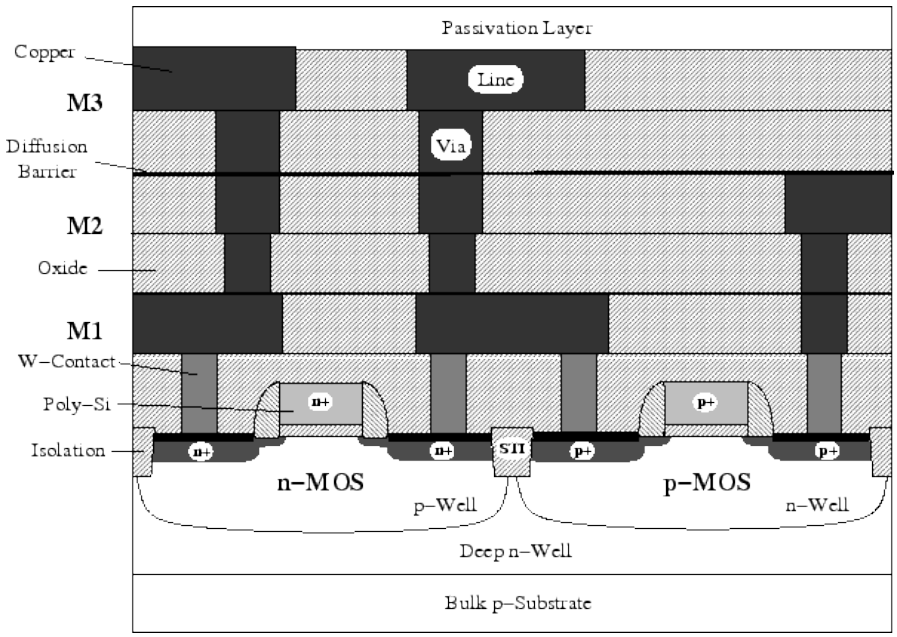

Figure 1.1: Schematic section view of an IC (Wittmann [69]). From bottom to top, we have: the substrate and the active layer with the transistors, *i.e.* front-end-of-line (FEOL); the metal layers (M1, M2, M3), *i.e.* the back-end-of-line (BEOL).

Beyond the 28nm technology node, the cost per transistor has increased for the first time (figure 1.2). Not only are the transistors more expensive to manufacture, but the thinner wires also implies more intricate BEOL layers patterning, further increasing the cost. Since then, it kept on getting more expensive and less cost effective. Yes, the industry still manages to cram more and more transistors on the same surface of silicon, but its profitability is stagnating if not crumbling.

But money is not the only limitation in this situation. Even assuming the technology can keep up with the scaling, and the International Technology Roadmap for Semiconductors (ITRS)

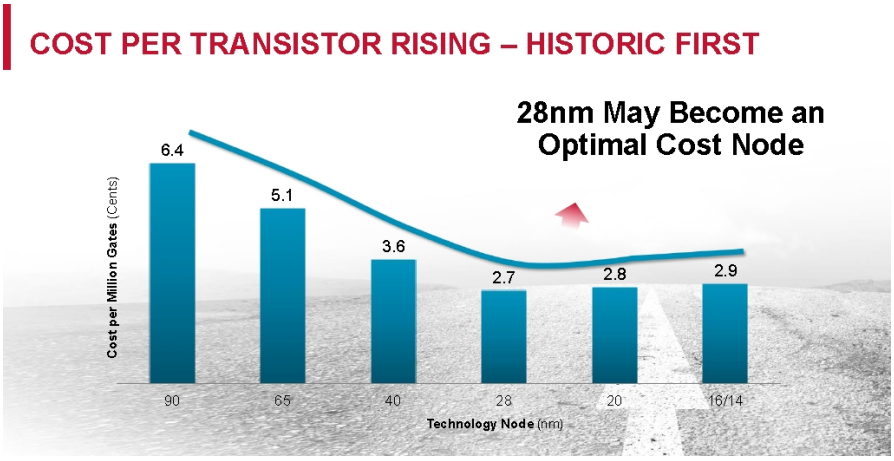

Figure 1.2: Cost of transistors per technology node ([56]).

expects the 5nm node by 2020, another problem arises. With the shrinking of transistors, the gate delay decreases, but the wire delay increases (see figure 1.3). The wire delay is already dominant since the 180nm node in 1999, but beyond 16nm, it is ten thousandfold more.

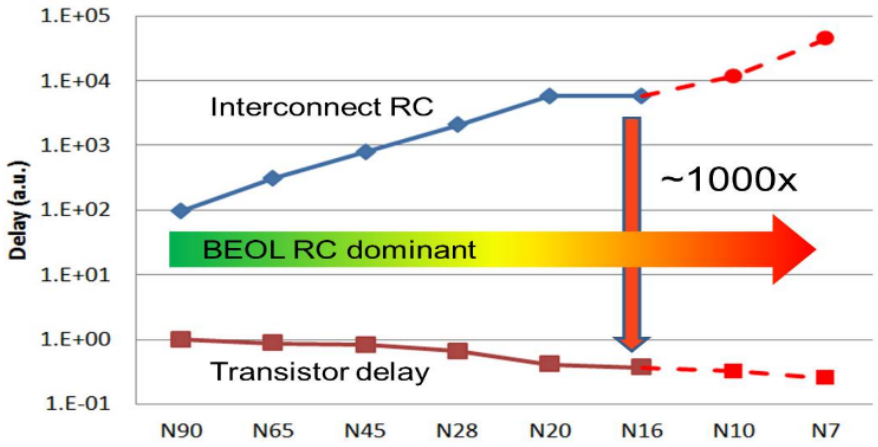

Figure 1.3: Interconnect RC (wire) delay versus Transistor (gate) delay ([71]).

To circumvent this increasing delay, engineers need to insert repeaters and buffers in the design, which occupy precious space on the die. The ITRS is aware of this situation. That is why in 2014, its committee reorganized the working groups into seven focus topics. Some of them, such as "System Integration" and "Heterogeneous Integration", have for mission to explore and develop 3D integration technologies, *i.e.* stacking several layers of transistors, effectively increasing the number of cells per unit area.

3D ICs can help save the day by reducing the area, improving the yield and casting aside the need for repeaters and buffers; once the dies are stacked and the length reduced, there won't be any need for them. Indeed, as logical block are stacked and brought closer to each other, their interconnection is shorter (see figure 1.4). The aim of system partitioning is to shorten the longest nets and preserve the shortest.

However, 3D is not around the corner just yet. Amongst other challenges, the industry still lacks proven toolchains capable of designing those 3D ICs. This is where we intervene: by intending to develop a complete toolchain automating the design and partitioning of 3D ICs.

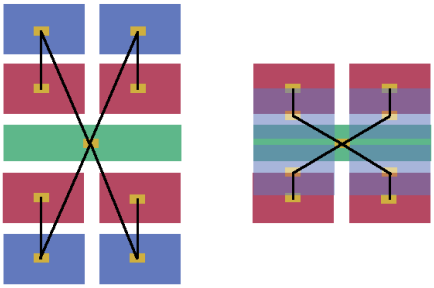

Figure 1.4: Transformation from a 2D to a 3D architecture. As the dies are stacked, the farthest blocks are brought closer and the interconnections are shortened.

In this report, we will see what is the scope of this subject (chapter 2), present some of the challenges 3D ICs face (chapter 3), explain what we developed to contribute to the research effort (chapter 4), discuss some preliminary results (chapter 5) and take a peek behind the door of the future of this project (chapter 6).

Chapter 2

State of the Art: How do we go 3D?

In this chapter, we will first define the basic terms (section 2.1) needed to discuss partitioning (section 2.2). Said discussion will focus on some of the most popular partitioning algorithms, their implementation and why hypergraphs are used. In chapter 4, we will present `PHONEY`, our software implementing those methods.

Next, we will present some physical integration techniques (section 2.3) and existing design flows (section 2.4).

2.1 Definitions

The following section will define the basic terms and concepts used in this work.

Definition 1: Hypergraph

> A *hypergraph* $H = (V, E)$ is a set of vertices V and a set of hyperedges E. Each hyperedge is a subset of vertices and its size is the cardinality of this subset. Each vertex v and hyperedge e is associated with a weight $w(v)$ and $w(e)$, respectively.

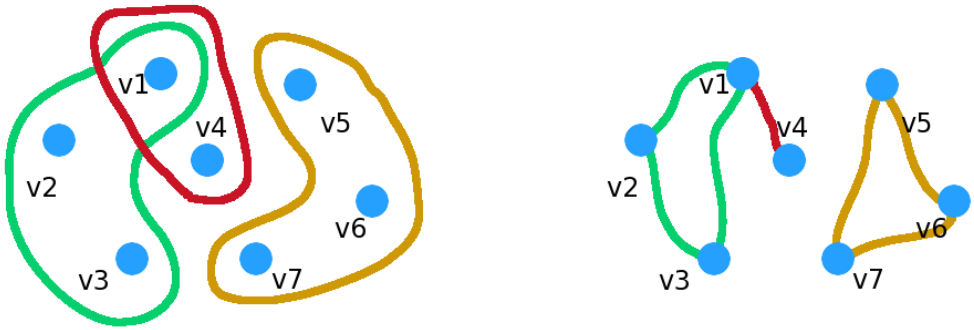

Figure 2.1: On the left, a hypergraph and on the right, the corresponding graph.

Definition 2: k-way partitioning

> A *k-way partitioning* is a decomposition of V into k disjoint subsets $V_1, V_2, ..., V_k$, such that $\bigcup_i V_i = V$. Each of these k subsets is called a partition.
>
> When $k = 2$, the decomposition is called a bipartitioning.

Definition 3: Cutsize

> The *cutsize* of a k-way partitioning is the sum of the weights of the hyperedges spanning several partitions, *i.e.* hyperedges containing vertices from different partitions.

Definition 4: Balanced partitioning

> A partitioning is balanced with respect to a constraint $[l, u]$, with $l < u$, when for each partition V_i, $l \leq \sum_{v \in V_i} w(v) \leq u$.

Definition 5: Balanced partitioning problem

> Given a hypergraph G and a balancing constraint $[l, u]$, the balanced partitioning problem is to compute a k-way partitioning with the minimal cutsize and respecting the balancing constraint.

2.2 The Usual Suspects – Partitioning algorithms

As the problem of graph partitioning is NP-complete [18], it needs heuristics to be solved. Typical algorithms include local search based solutions (such as Kernighan-Lin (KL) [32] and Fiduccia-Mattheyses (FM) [17]), simulated annealing [25], tabu search [50], etc. To scale up with the problem size and handle millions of nodes, the multilevel paradigm was developed. Nowadays, as we reach graphs with billions of nodes, new methods need to be researched ([65, 68]).

In this section, we present some of the most popular partitioning methods.

2.2.1 Kernighan-Lin Algorithm

Kernighan-Lin (KL) is a partitioning method published by Kernighan and Lin [32], based on gain cell swap. The idea is to randomly bipartition an undirected graph and improve its cutsize by swapping pairs of nodes across the partition.

During each pass of the algorithm, we compute the cost of each cell in both partitions P_p, with $p \in \{1, 2\}$:

- External cost of cell x belonging to partition P_1:

$$E_x = \sum_{i \in P_2} c(x, i)$$

  with $c(x, i)$ the weight of the edge between the nodes x and i.

- Internal cost of cell x belonging to partition P_1:

$$I_x = \sum_{i \in P_1} c(x, i)$$

From these two costs, we can compute the gain of swapping the cells x and y:

$$gain(x, y) = (E_x - I_x) + (E_y - I_y) - 2 \cdot c(x, y)$$

At each step of the pass, we (i) compute the gain of all unlocked nodes, (ii) swap the pair with the highest gain and lock both nodes into their new partition, (iii) record the current cutsize.

When all pairs have been swapped and if the cutsize has not decreased, we stop. Otherwise, we unlock all the nodes and start a new pass using the m first swaps leading to the minimal cutsize.

A limitation that may seem obvious is that the vertices need to have the same weight if we want the bipartition to be balanced. There is however a workaround (Kahng et al. [27, p. 41]). We first define a unit-cost weight W_u that is the greatest common divisor of all node weights W_n. Then, we instantiate all nodes W_n/W_u times with a weight W_u and link those instances together using infinite-weight edges. Those edges will become high priority whenever a portion of the node has crossed the partition boundary, forcing KL to keep the instances together.

Application to circuits In order to obtain an undirected weighted graph G from the circuit, we can use the k-clique model: Each net that contains k gates becomes a k-clique in G. In a k-clique, each node can have a weight of $\frac{1}{k-1}$, and if an edge already exists, its weight is simply summed instead of being duplicated.

Time complexity We need $\mathcal{O}(n^2)$ to update the gain of n nodes, and $\mathcal{O}(n^3)$ to compare the $\mathcal{O}(n^2)$ pairs and choose the maximum gain. This leads to a $\mathcal{O}(n^3)$ time complexity. However, if the nodes are sorted and the comparison cleverly ordered, it can drop to $\mathcal{O}(n^2 \log n)$ [14, 16].

2.2.2 Fiduccia-Mattheyses Algorithm

Fiduccia-Mattheyses (FM) is a method to bipartition hypergraphs, developed by Fiduccia and Mattheyses [17]. In contrast with KL, FM is based on cell moves instead of cell swaps. There is then no need to compute lots of gains, but a balance constraint becomes necessary to ensure that a cell move does not disbalance the bipartition.

Like for KL, the first bipartition is random and will be improved upon successive passes. At the beginning of the first pass, we compute the gain of each cell x:

$$gain(x) = FS(x) - TE(x)$$

with $FS(x)$ the number of nets having x as its only cell in P_1 and $TE(x)$ the number of nets containing x and being entirely located in P_1.

At each step of the pass, we (i) choose a free cell with maximum gain which move is allowed by the balance constraint, (ii) move it and lock it, (iii) update the gain of the impacted cells and (iv) record the current cutsize.

When the cutsize is not improved at the end of a pass, we stop. Otherwise, we keep the fist m moves that led to the minimum cutsize and begin a new pass.

Application to circuits As FM already works on hypergraphs, the application to circuits is fairly immediate: Each net becomes an hyperedge and each cell becomes a vertex.

Time complexity Since we only need to update the gain of cell movement for the cells impacted by previous moves, the time complexity drops to $\mathcal{O}(n)$, which is a huge improvement compared to the $\mathcal{O}(n^3)$ of KL.

2.2.3 EIG Algorithm

Hagen and Kahng [21] showed that the second smallest eigenvalue of the connectivity matrix of the graph is a tight lower bound on the ratio cut metric. That metric is defined as $\frac{c(X,Y)}{|X||Y|}$ with $c(X,Y)$ the cutsize between the two partitions.

Based on this result, they built an algorithm to make bipartition of graphs with minimal ratio cut.

The idea is first to approximate the hypergraph with an undirected graph using a k-clique model, similar to what was done for KL. From this graph, we derive the degree matrix D where d_{ii} is the sum of all the weights of the incident edges on node i. Similarly, we build the adjacent matrix A where a_{ij} gives the weight of the edge (i,j). The Laplacian matrix Q is then $Q = D - A$, from which we extract the eigenvector and its second smallest value. Using the ordered eigenvector, we then compute $n-1$ partitionings:

$$(\{v_1\},\{v_2,...,v_n\}),(\{v_1,v_2\},\{v_3,...,v_n\}),...,(\{v_1,...,v_{n-1}\},\{v_n\})$$

Although this method is not implemented in traditional libraries, it may be useful to set a lower bound on the ratio cut.

2.2.4 Other heuristics

In parallel with the aforementioned methods, there exist other heuristics in competition. Some are briefly presented here for the sake of completeness, but as they have been supplanted by other paradigms along the years for circuit partitioning, we will not dive too deep into the details.

Local Search Strategy A local search will look in the neighborhood of the current solution for a better one. If there is nothing better in the direct vicinity of the position, just stop. As Pirlot [50] says, this strategy is also known as "descent" strategy and is basically what was done in `PHONEY` when we tried to disbalance the energy distribution in the partitions, while keeping the area balanced (see sections 4.1 and 5.2).

The problem of this strategy is that it will get stuck in local minima. Escaping requires a deterioration of the solution, which is done by Simulated Annealing (SA) and Tabu Search (TS).

Simulated Annealing Pirlot [50] presents this heuristic like the evolution of a local search. Instead of looking in the neighborhood of the current solution, we pick one randomly. If it is better, keep it, otherwise we have a probability p to keep it and $(1-p)$ to discard it. p is usually a Boltzmann-like distribution:

$$p(n) = \exp\left(-\frac{1}{T(n)}\Delta F_n\right)$$

with n the current step, $\Delta F_n = F(x) - F(x_n)$ the distance from the current solution, and $T(n)$ is what we could call the "temperature" by analogy with the annealing of steel. Each L steps, the temperature decreases, lowering the probability to accept a worse solution.

The stopping condition can be a number of steps from the beginning or a number of steps without enough improvement of the solution.

Tabu Search A tabu search, as its name and Pirlot [50] indicate, is based on a tabu of some sort. We will proceed like for a regular local search, the difference being that we establish a tabu list that prevents from cycling through the same solutions. In this list, we could simply store the L last visited solutions, forbidding them. But this is still not good enough and performs poorly in practice. Another approach is the forbid *movements*, *e.g.* changing some coordinate from 0 to 1. However, should a potential solution forbidden by the tabu list be so good according to some "aspiration level", the tabu can be overlooked and the solution be chosen.

2.2.5 Multilevel paradigm

As the number of cells grows, the graphs become more difficult to partition in a practical time. In order to use proven algorithms on larger graphs, they need to be scaled down. This is what the multilevel framework has been developed for [29]: (i) coarsen the graph, (ii) partition it using aforementioned algorithms and (iii) uncoarsen the graph while refining the partitioning.

Compared to a standard partitioning algorithm alone, Karypis and Kumar [28] proved that the multilevel implementation yields a worse cutsize by at most a small factor.

Coarsening By coarsening the graph, we want it to become smaller in order to ease the job of the partitioning algorithm. Hereunder we present some of the most popular steps used in the hMetis package [30], a multilevel hypergraph partitioning implementation (see section 2.2.6).

- Edge Coarsening (EC): The aim is to match pairs of vertices connected by many hyperedges. First, each hyperedge e of size $|e|$ is assigned a new edge-weight of $1/(|e|-1)$, that may be different from their hyperedge weight. Then, as each vertex v is visited in a random order, all unmatched vertices u belonging to hyperedges containing v are considered. For each candidate, we compute the weight of the edge connecting u and v, that is the sum of all the edge-weights of the hyperedges containing u and v. The pair with the maximum weight wins and gets merged.

  This is similar to the scheme that converts the hypergraph into a graph, but this is done implicitly. And this is why the coarsening using this technique is quite slow: only pairs of vertices are matched at each pass.

- Hyperedge Coarsening (HEC): This time, we contract all the vertices belonging to the same hyperedge together at once. First, the hyperedges are sorted in a decreasing hyperedge weight order. If several hyperedges have the same weight, they are sorted in an increasing size order. Hyperedges are then visited in that order and for each hyperedge containing vertices that have not been matched yet, all its vertices are collapsed together.

  However, during each pass, a majority of the hyperedges don't get contracted because they contain vertices that have already been matched. The main problem is thus that some vertices remain unmatched and see their weight become less significant relative to merged vertices, distorting the shape of the coarser hypergraph.

- Modified Hyperedge Coarsening (MHEC): This last scheme adds a new step after HEC. At the end of each step, when all vertices that could be matched have been matched, the list of hyperedges is traversed again. Then, for each hyperedge that has not been contracted yet, its vertices that do not belong to contracted hyperedges are merged together.

  This trick avoid orphan vertices and preserve the shape of the hypergraph.

Many other schemes exist, have been developed and improved upon for the past 20 years [3, 29, 37, 41].

Partitioning At this step, any partitioning algorithm presented in this section can be used. However, as this coarsened hypergraph as yet to be uncoarsened, it is not necessary to find the optimal partitioning. A good-enough strategy is to compute a random bisection (in the case of a bipartitioning) and let the refinement phase improve it.

Another strategy is to generate n partitions at the coarsest step and refine them all in parallel. Then after each refinement pass, we can drop the worst partition and keep only one at the end. For example, Karypis et al. [30] used ten initial partitions and dropped the 10% worst of them after each pass, leading to a 3 to 4% reduction of the cut.

Uncoarsening and refinement This last step undo what has been done in the first phase while maintaining and improving the partition obtained in the second phase. If the uncoarsening is pretty straightforward, the refinement can take different forms. The most basic implementation is simply to use the partitioning algorithms presented in this section and let them do their work. However, since the partitioning needs to be done several times and already starts with a good cutsize, some improvements can be made.

- We can limit the number of passes of the partitioner during each pass of the refinement.
- We can stop the algorithm as soon as the cutsize has not improved after n vertices have been moved.
- Instead of moving one vertex at a time (like in FM), we can move the whole hyperedge across the boundary as long as the balance constraint is met.

2.2.6 Implementations

Numerous software libraries implement some flavor of (hyper)graph partitioning. We will present some of the most popular in this section. The reader can refer to [63] for a more extensive survey.

- Chaco[1] (Hendrickson and Leland [22]): One of the first package implementing several state-of-the-art algorithms at the time. On top of implementing raw KL and its multilevel variant, it also proposes simple, spectral and inertial partitioning algorithms. Although it does not work on hypergraphs nor does it fundamentally modify or improve basic algorithms, it has the advantage to offer a stable ground for further development and more sophisticated implementations comparison.
- hMETIS[2] (Karypis et al. [30]): A fork of the METIS package from the same team and uses mainly the techniques presented in section 2.2.5. Even though the algorithms used are dating and it has not been maintained for 10 years (the last alpha version is from may 2007), it is still used as a reference benchmark.
- PaToH[3] (Partitioning Tools for Hypergraphs, Çatalyürek and Aykanat [7]): Multilevel implementation using a not-so-fancy matching-based coarsening and slight modifications of

[1] `https://cfwebprod.sandia.gov/cfdocs/CompResearch/templates/insert/softwre.cfm?sw=36`
[2] `http://glaros.dtc.umn.edu/gkhome/metis/hmetis/overview`
[3] `http://bmi.osu.edu/umit/software.html#patoh`

KL and FM for uncoarsening. The more interesting bit is their partitioning phase; they use a greedy hypergraph growing scheme. It selects first a few random vertices, then grows a partition around them by adding unselected vertices in order of their FM gain, until the balance constraint is met.

- MLPart[4] (Caldwell et al. [6]): A multilevel implemetation that was develop with the obsession of besting hMETIS, which it does in some cases and fails in others. Its coarsening phase uses a modified EC with some particular attributes. For the initial partitioning, it randomly assigns nodes to each partition *w.r.t.* the balance constraint, in decreasing order of their size. Finally, the uncoarsening is simply an improved FM.

- Par*k*way (Trifunovic and Knottenbelt [64]): A parallel implementation of the multilevel scheme for hypergraphs. The serial multilevel scheme on which it is based uses classical methods for the coarsening phase and initial partitioning phase, and FM or some declination for the uncoarsening phase. Its particularity is the parallelization of this serial execution using clever data distribution and communication amongst the processors (*e.g.* exchanging matching requests in the coarsening phase and moving sets of vertices in the uncoarsening phase).

  Note that METIS also has a parallel variant in ParMETIS, but which only works on graphs.

- Zoltan[5] (Devine et al. [15]): Another parallel implementation of the multilevel scheme. Of its own team acknowledgment, it is in direct competition with Par*k*way. They share the same principles on several points, execpt some details. For example, Zoltan's initial partitioning tasks each processor to compute a random partition and keeps the global best as a start for the uncoarsening. In the end, Zoltan is a more straightforward parallelization than Par*k*way, with less communication between the nodes and less fancy computations. This probably explains why, compared to Par*k*way, Zoltan's cutsize of the hypergraph is rarely better, but the partitioning time is far shorter for the data sets presented in the founding article.

- FEHG (Lotfifar and Johnson [37]) : More recent implementation of multilevel scheme (2015), but very similar to PaToH. The only difference is in the coarsening algorithm which relies on rough set clustering. This algorithm allows highlighting patterns in a graph by measuring the similarity of nodes using the Jaccard index.

hMETIS and PaToH are the cornerstone implementations used in this work, but we want to extend our support to other libraries such as MLPart and Scotch. A limiting factor is the documentation available, their compatibility with our development platform and the mere existence of a full library.

2.2.7 Why hypergraphs?

All hypergraphs can be represented as regular graphs, and lots of tools already exist to handle graphs. When we want to express the wire length between two nodes, although it can easily be done using graph's edge weights, it is not expressable as a hypergraph's hyperedge weight, the hyperedge linking several nodes all connected with different wire lengths.

[4] `http://vlsicad.ucsd.edu/GSRC/bookshelf/Slots/Partitioning/MLPart/`
[5] `http://www.cs.sandia.gov/zoltan/`

However, when tackling the partitioning problem, working with hypergraphs gives the vertices belonging to the same hyperedge a coherence. In the context of 3D ICs, we work with buses connecting blocks of gates, and we want them to be handled as such.

Moreover, Ihler et al. [23] demonstrated that the mincut partition obtained for a circuit is not as accurate as would be a hypergraph's. More precisely, they define a *cut-model* in definitions 6 and 7, and show that there is no such thing in general if only positive weights are allowed (which is the case when studying circuits).

Definition 6: Cut-model

> An edge-weighted graph (V, E) is a cut-model for an edge-weighted hypergraph (V, H) if the weight of the edges cut by any bipartition of V in the graph is the same as the weight of the hyperedges cut by the same bipartition in the hypergraph.

A more formal way to define the principle is as follows :

Definition 7: Cut-model and mincut-model

> A graph (V, E) on k vertices is a cut-model (for a unit weight hyperedge on k vertices) if the weight of any cut induced by a non-empty proper subset W of V is equal to one.
>
> A graph $(V \cup D, E)$ on $k + d$ vertices is called a min-cut-model (for a unit weight hyperedge on k vertices) if for every non-empty subset W of V we have that the weight of any cut with minimum weight (mincut) under those separating W from $V \setminus W$ is equal to one. It must be zero fo $W = \emptyset$.

2.3 Monolithic vs stacked – Integration flavors

In the 3D-IC world, two main philosophies coexist. There is not a single integration system dominating the others and attracting all the attention. Actually, there exist two main ways to assemble multiple layers of transistors: as a monolithic bloc [61, chap. 2] or stacking "standard" layers[61].

Monolithic 3D ICs There exist several levels of monolithic integration (see fig 2.2): transistor- [35, 52], gate- [47, 46] and block-level [45].

The transistor-level integration can separate the transistors of standard cells on different layers of the 3D IC, enabling the finest grain integration. However, it would require the redesign of standard cells.

The next grain level is to work on gates. Existing standard cells can be reused, but the partitioning algorithm needs to handle millions of nodes.

Finally, monolithic 3D chips can be designed from functional blocks. On top of being able to reuse existing standard cells, even IPs could be reused as such. Yet, this level of integration does not take full advantage of the fine integration nature of monolithic 3D.

Monolithic has drawbacks, though. Its sequential manufacturing is expensive and most of the wire length gained from shorter 2D connections is lost on 3D connections due to all the small nets that are cut by the partitioner. What is worst is that the wire capacitance is even increasing due to the multiplication of nets.

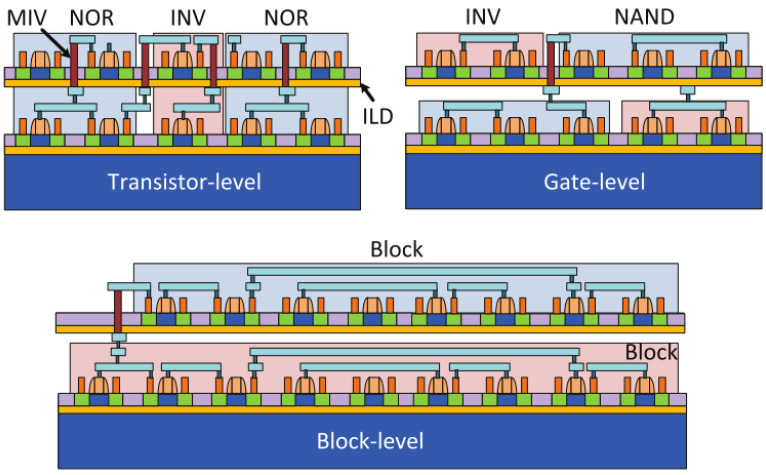

Figure 2.2: The three levels of monolithic 3D ICs: transistor-, gate- and block-level (Panth et al. [47]).

3D stacking When stacking ICs to make them 3D, each layer is a full fledged IC, with its substrate and metal layers. To bond them together, two main methods exist:

- face-to-face (F2F), using F2F vias ([49]). We put the BEOL of each chip in front of each other (see figure 2.3). Although this method avoids drilling through the substrate of the dies, it is limited to two layers.
- face-to-back (F2B), using µ-bumps and through-silicon-vias (TSVs). The BEOL of one die is stacked to the FEOL of the other, this time allowing unlimited layers to be stacked (see figure 2.4).

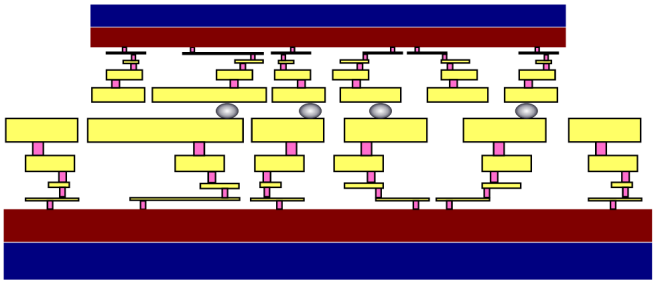

Figure 2.3: Face-to-face integration, the two front-end-of-line are facing each other.

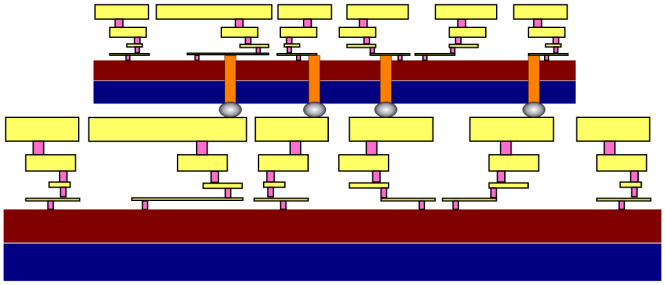

Figure 2.4: Face-to-back integration, the µ-bumps mounted on the BEOL of the bottom tier are aligned with the TSV in the FEOL of the top tier.

Furthermore, there are three other methods to bond wafers together:

- wafer-to-wafer (W2W), simply stack two full wafers. Easier for high-yield processes with wafers of the same size.
- die-to-wafer (D2W), bond individual dies on a wafer. More complex integration, but gives a better yield (failed dies are discarded) and does not require die or wafers of the same size.
- die-to-die (D2D), bond individual dies together. Further complicates the manufacturing process and cost due to pick-and-place operations, but the yield is further improved.

The research on all families of methods is not restrained to a few research groups. Lots of them are heavily tied with the industry and supported by public grants. In Europe only, FP7 and H2020 project frameworks have supported or are still supporting numerous teams.

FP7 Projects:

- FAB2ASM: *Efficient and Precise 3D Integration of Heterogeneous Microsystems from Fabrication to Assembly*[6].
- JEMSIP_3D: *Joint Equipment and Materials for System-in-Package and 3D-Integration*[7].
- NANOPACK: *Nano Packaging Technology for Interconnect and Heat Dissipation*[8].
- ELITE: *Extended Large (3D) Integration TEchnology*[9].

Horizon 2020 projects:

- TAKE5: *Technology Advances and Key Enablers for 5 nm*[10]. This projects aims at the development of the 5nm node. Even though scaling is worth exploring, the 3D integration path is not to be left aside.
- NeuRAM3: *NEUral computing aRchitectures in Advanced Monolithic 3D-VLSI nano-technologies*[11].
- METRO4-3D: *Metrology for future 3D-technologies*[12].

2.4 3D IC design flow

As the research in 3D IC partitioning and manufacturing advanced, it makes sense that comprehensive 3D workflows have been developed to prototype the innovations. Here follows a selected list of some academic integration flows.

[6] `http://cordis.europa.eu/project/rcn/94309_en.html`
[7] `http://cordis.europa.eu/project/rcn/201936_en.html`
[8] `http://cordis.europa.eu/project/rcn/85245_en.html`
[9] `http://cordis.europa.eu/project/rcn/85238_en.html`
[10] `http://cordis.europa.eu/project/rcn/203403_en.html`
[11] `http://cordis.europa.eu/project/rcn/199168_en.html`
[12] `http://cordis.europa.eu/project/rcn/199865_en.html`

CoolCube In the past few years, the CEA LETI has been working on a new way to integrate transistors in 3D: [4, 5, 9, 39, 67]. Their goal is to manufacture 3D monolithic chips using innovative stacking techniques. However, they still face the problem of partitioning and gates repartition on the tiers.

Shrunk2D Another and slightly older monolithic 3D IC (M3D) toolchain that shrinks the cells to evaluate the wire length and placement in 3D [47]. By shrinking the dimensions by a factor $1/\sqrt{2}$ everything is twice as small. After placing and routing the shrunk cells, they are expanded back to their original size and every overlapping cells are separated on two different dies. This evaluation of course only works when we assume the breadth of the vertical integration to be negligible, which is a fair assumption for M3D.

Cascade2D An M3D design flow developed by Chang et al. [8] as an improvement over Shrunk2D. Some of its enhancements compared to its predecessor includes the addition of a monolithic inter-tier via (MIV) planning stage between the partitioning and 3D placement, the versatility of compatible partitioning algorithms and the ability to work at block-level.

As promising as it is, Cascade2D sadly is highly interlaced with Cadence® Innovus™, limiting its reusability without the proper licenses.

MEVA-3D is a bit different from the previous tools: it is an automated design flow evaluation designed by Cong et al. [12]. The team overlooks most of the 3D design to focus on the thermal dissipation and comparison between 2D and 3D architectures. They spend most of their effort on thermal network modeling as well as thermal via insertion in the design to ensure a correct heat flow through the layers.

Chapter 3

Challenges: Why is it not done yet?

3D is not a novel idea. In 1979, Geis et al. [19] already demonstrated the concept. They were followed on the 80's [1, 10, 20, 31, 34, 42, 44, 60, 62] and the 90's [43, 53, 58, 59] by a lot of theoretical research on 3D CMOS and early integration. But it is only in the 2000's [57] that full workflows began to be developed and full chips manufacturing began to be considered.

The main factor why it has not been more thoroughly investigated earlier is that the cost to scale down the technology was simply lower than developing 3D integration technologies. Hence, as attractive a research subject it was, the industry was simply not ready to heavely invest in it.

Apart from the pure manufacturing management, multiple issues arise when we tackle the 3D integration problem. Some of those problems are exposed in this section to alert the reader to the complexity and the span of the situation.

3.1 Manufacturing

As we already discussed in section 2.3, there exist multiple ways to manufacture a 3D IC. Xu et al. [70] did a review of yield enhancement for stacked 3D IC and identified two main sources of yield loss when going 3D:

- Stack yield loss. When stacking two dies using the W2W technique, you may stack a bad die with a good one. This issue is referred to as the "known good die" problem.
- Assembly yield loss. This is simply the extension of the traditional yield loss in 2D architectures: manufacturing is a risky process and may fail. 3D stacking further worsen the situation by adding extra steps such as wafers alignment and handling, TSV defects and interactions with the package. The temperature, signal integrity and reliability of 3D ICs are affected by how the dies are stacked and where the µ-bumps are placed [66, 26]. Indeed, due to the difference in coefficient of thermal expansion between the substrate and package, when the 3D IC undergo thermal load, residual stress can get transferred from the package to the substrate, degrading its reliability.

Techniques to reduce those losses respectively include pre-bond die testing and TSV redundancy to alleviate the defective ones. However, the precise loss in yield is still unsure. Assuming the number of defaults per unit area is constant, manufacturing smaller dies will increase the yield, but the 3D integration process may reduce the yield at the same time. Whether both factors counterbalance is part of the challenge.

3.2 Thermally-aware Design

Manufacturing 3D IC efficiently is one thing, but having them function properly without failing is another entirely. As the dies are stacked, the heat generated by the standard cells becomes increasingly difficult to dissipate and the reliability of the chip decreases [36]. Farther the cells are placed from the heatsink, the more critical they get.

In the recent years, two research directions have emerged to address this problem:

- Thermal co-design [11, 13, 40, 51, 72]: either by cleverly spreading the cells on the dies to limit the hotspots and heat gradient, or by using thermal TSVs whose role is to canalize and improve the heat flow through the layers. For example, by forcing the high-power cells on the layer closer to the heat sink, Athikulwongse et al. [2] outperforms state-of-the-art temperature aware placers by 10% and 33% in terms of maximum temperature and temperature difference.
- Micro-fluidic cooling [33, 54, 55]: leave space between the layers and use some deionized fluid to evacuate the heat channeled through inter-tiears fins. Advocates of this technique argue that micro-fluidic cooling improves performance and efficiency of a given architecture compared to traditional air cooling. If it seems promising on paper, it is a complex technology to put into practice, as it further complicates the manufacturing and needs a paradigm shift of chips integration in complete systems.

3.3 Clock Tree Synthesis

When synthesizing a clock tree, we want to avoid clock skew (maximum difference in clock arrival between sinks) [38]. We could have separate clock I/Os on each layer to enable single-die clock testing and increasing the yield. Indeed, current clock tree synthesis (CTS) need the full bonded chip to be tested. If die CTS could be tested individually, even more failed dies could be discarded early in the manufacturing process.

3.4 Power Delivery Network

The power delivery network (PDN) also needs to be revised when going 3D. When the power needs of the IC scale with the area footprint and number of layers, the pins needed to deliver this power from an external source are still limited by the same footprint of the chip. The imbalance is a challenge [24].

What can be even more of a challenge is when we consider an air-cooled 3D IC. The power delivery pins and heatsink are on opposite sides of the chip, but the same type of cells need them the most at the same time. The most power consuming cells should be placed close to the power source in order to avoid a complex PDN, but at the same time they will dissipate the most heat and should be placed to the heatsink to avoid failures. Teams working on the problem are trying to optimize the power demand across layers, such as Pantht et al. [48] who developed a PDN aware gate-level partitioner.

Chapter 4

Work: Where are we now?

Nowadays, there is still not any commercial 3D IC design toolchain. Our aim is to bring an interface between existing 2D tools to enable automatic 3D integration, and reducing as much as possible the proprietary intermediaries, in order to keep the flow as open as possible. As we discussed in section 2.4, some academic toolchains are already pushed by different research teams around the globe. Their common divisor is their ties to specific, expensive tools and their lack of flexibility. This is a major drawback we want to alleviate in our work.

The figure 4.1 gives a macroscopic idea of the interface we aim at developing. The elements on the left-hand side are existing solution and tools, while the right-hand blocs are currently developed to automate what is currently mainly done by hand.

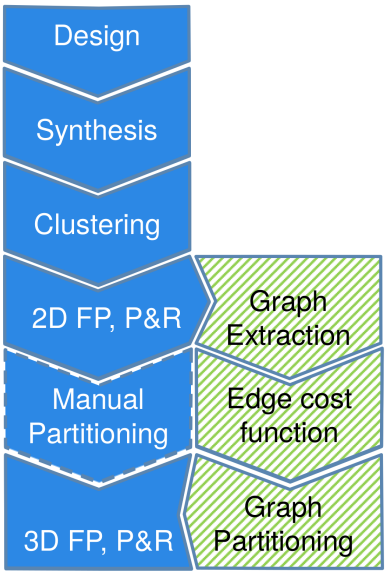

Figure 4.1: Typical 3D IC design flow. We want to bypass the usual manual partitioning with an automated graph extraction and partitioning.

In this section, we will present what has been developed so far and how those components interact with each other.

4.1 Graph Extraction

We developed `PHONEY`[1], a software tool extracting the hypergraph of a 2D placed and routed design, feeding it to a partitioner and formating the output for the Place and Route (P&R) tool to process back. The processing is pretty straightforward: each cluster of the design becomes a vertex and whenever a net is connecting cells belonging to different clusters, the net becomes an hyperedge. The (hyper)graph is then formated in such a way that the partitioner can do its job and generate nice and clean partitions. The partitioner in question can be any library with a command-line interface; all that is needed is one function translating our representation of a graph into one understandable by the library. So far, we support METIS, hMETIS and PaToH. The support for METIS was added to compare the partition quality between a hypergraph and its corresponding graph. Notwithstanding the motivation for hypergraphs we discussed in section 2.2.7, we did not note a significant difference in quality between the two approaches.

Here follows a summary of the main features of `METIS`:

- Extract the design as a graph or hypergraph.
- Choice of the partitioner: METIS, hMETIS or PaToH.
- Disbalance of the graph weights power-wise. This can either be done by inserting a dummy node or by manually overriding the partitioning and moving nodes around "by hand". Those are further discussed in section 5.2.
- Type of edge weights.
- The script and partitioner can be seeded, insuring repeatability of the results.

4.2 Stats extraction and interface with open standards

The extraction of statistics about the design is essential to the weights of the graph. Those can either be obtained through the analysis of a third-party software or from the output of the P&R tool.

In the early days of our flow, `PHONEY` was only able to read a proprietary tool-specific file formatting. Although the format was documented and consisted of standard text files, you needed to be aware of it if you wanted to use the script. With the addition of `def_parser`[2], our toolchain is now compliant with the open standards Design Exchange Format (DEF) and Library Exchange Format (LEF). This script feeds on a DEF file containing all the information on the placed and routed design, and on an associated LEF file containing the physical description of the standard cells of the specific technology used.

One of its role is to translate those standard files into an input that `PHONEY` is able to process. The other is to cluster and analyze the design.

Here follows a summary of the main features of `def_parser`:

- Compatibility with open standards.

[1] "Partitioning of Hypergraph Obviously Not EasY", `https://github.com/parastuffs/metis_unicorn/blob/master/script/phoney.py`

[2] `https://github.com/parastuffs/def_parser`

- Extract all the cells, pins and nets of the design and their interconnection.
- Analyze the geometry of the die.
- Design clustering (see section 4.3 and analysis).

4.3 Naive Clustering

The aim of our clustering is manyfold: (i) reduce the order of the graph for the partitioner to have an easier job, (ii) slice the die and analyze the interconnectivity between the clusters, (iii) be a proof-of-concept clustering that allows us to already prepare the extraction of statistics for more complex clustering methods that may replace this one.

The way it is implemented is simplistic and naive: we slice the die into a grid with each cell being a cluster. We consider that the P&R has placed close to each other standard cells that participate to the same logical function, hence making them a sound logical block that can be used as a cluster. By playing on the granularity of the clustering, we hope to highlight the impact of inter-cluster wire length when tackling the gate- or block-level dilemma.

As we will see in chapter 5, this geometrical clustering also allows us to proceed to a first, rough study of the interconnection of standard cells in a given design.

Chapter 5

Results: What can we say?

In this chapter, we will analyze the results of some partitioning using `PHONEY` and hMETIS as well as some clustering computed by `def_parser`. We focus on wire length (WL) reduction (section 5.1), power asymmetry (section 5.2), and inter- and intra-clustering connectivity (sections 5.3 and 5.4).

5.1 Wire length reduction

For the sections 5.1 and 5.2, we used three different sub-modules of the OpenSPARC T2, each with different level of wire or gate dominance (see table 5.1).

SPC (Core)	CCX (Crossbar)	RTX (Ethernet)
Gate dominated	Wire dominated	Balanced

Table 5.1: The three modules tested with our partitioning flow, each with a different level of gate or wire dominance.

Using `PHONEY`, we extracted the following weights for our graphs:

- Edges:
  - Number of wires per connection
  - Total wire length of the hyperedge
  - Average wire length of the nets in the hyperedge
  - Linear combination and inverse of those
- Vertices:
  - Area
  - Power

Note that if the partitioner supports multiple weights at the same time for the vertices, the edges can only be characterized with a single weight.

Our first objective was to find out the potential of our workflow by studying the WL reduction when going 3D. The figure 5.1 summarizes two reductions: the total WL and the longest net shortening.

We could have expected RTX to yield results between SPC and CCX due to its balanced design. However, whilst there are some gains, the median is worse than SPC. A more interesting

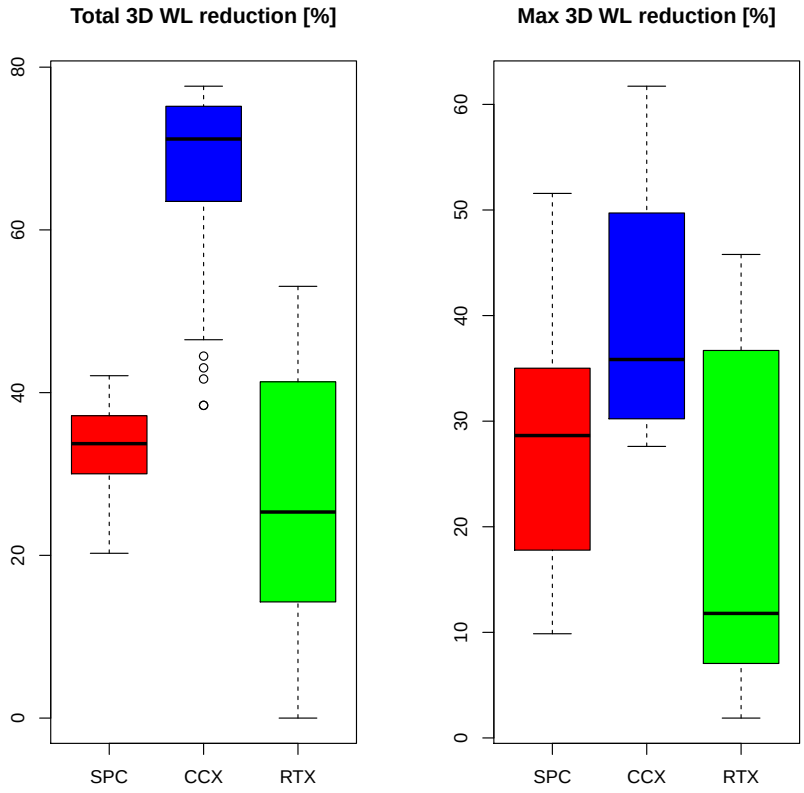

Figure 5.1: Total and maximum wire length reduction when going 3D.

result about RTX is that its set of solutions spans from 0 to 53% WL total reduction, showing that for some designs, the choice of appropriate metrics is critical. At the same time, the difference between the first and third quartile of SPC is only around 7%. Contrarily to RTX, other designs do not depend heavily on the metric, but this seems to be more of a corner case.

All in all, the quality of the partitioning depends on the weights used on the hypergraph and should not be picked lightly. Following our experiments, wire-dominated designs such as CCX can hope for up to 77% (median 71%) of wire length reduction and 61% (median 35%) of longest net reduction.

5.2 Asymmetric partitions

One of the limitations of popular implementations is their mania with balanced partitioning. For example, hMetis allows the user to set loose balance constraints, but will still try to produce partitions as balanced as possible. If we want to produce asymmetric (*i.e.* unbalanced) partitions, we need to trick the partitioner.

Say we want an asymmetric repartition of power on two dies of the same size. This gives two balance objectives the partitioner needs to follow. To unbalance only one of them, we can create a fake, dummy node with a fixed power density and a negligible area. Doing so, the partitioner will still balance both metrics, but the actual power balance will be offset. For example, a $70/30$ imbalance can be achieved by creating a dummy node with a power equal to 40% of the total power. The partitioner is then fooled into thinking it needs to balance 140% of this metric, from

which the dummy node will ultimately be removed.

By using the fixed vertices feature that most libraries implement, we can then choose the repartition of the asymmetry.

In practice, we tested this idea on the SPC module and obtained the results on figure 5.2.

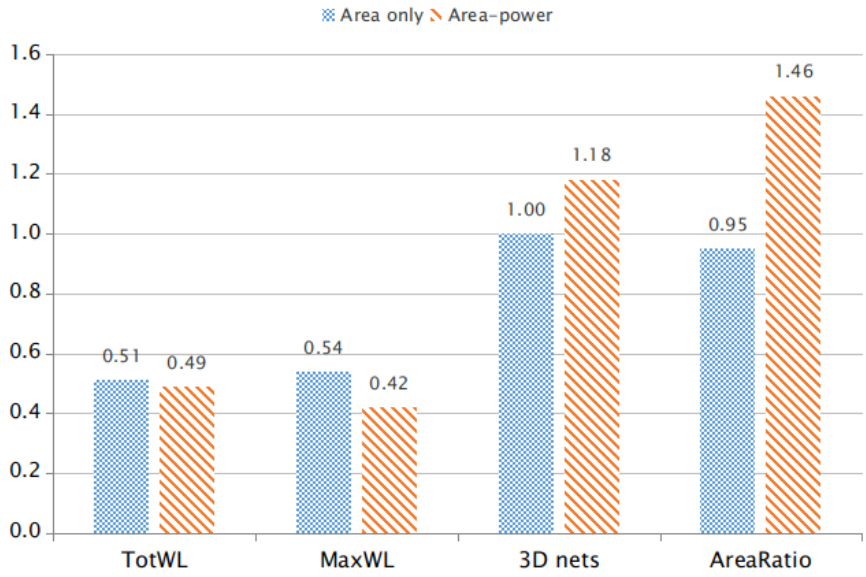

Figure 5.2: By forcing the asymmetry power-wise, we further gain some total wire length shorten the longest net, but there are globally more 3D nets and the aspect ratio is unbalanced. All values are the ratio between the 3D and 2D implementation.

In the end, although the $70/30$ power disbalanced objective is reached, the area balanced could not be maintained and drops to $60/40$. By using a less aggressive asymmetry objective, we were able to conserve the area balance, but with close to no gain to the wire length. Still, even though the two dies do not have the same size anymore, the disbalance allowed us to further reduce the total wire length by 2% and the maximum wire length by 12%, to the cost of an extra 18% of 3D nets.

5.3 Inter-cluster connectivity

To test our clustering and analyze it, we used some new designs at two different technology nodes (see table 5.2).

Design@tech	Cells	Nets	Total WL [μm]	Area [μm^2]
SmallBoom@7nm	121 580	137 171	308 947	10 401
Flipr@7nm	220 587	234 373	767 365	14 713
LDPC@7nm	42 471	49 633	173 167	3 309
SPC@45nm	289 812	306 118	10 112 735	906 827
CCX@45nm	185 777	200 999	5 455 493	313 464

Table 5.2: List of all tested architectures: Either using 45nm or 7nm standard cells library.

The aim of this experiment is to highlight similarities between the designs and try to find

tendencies of the impact of clustering on interconnection and wire length. The results are summarized on figures 5.3, 5.4, 5.5 and 5.6.

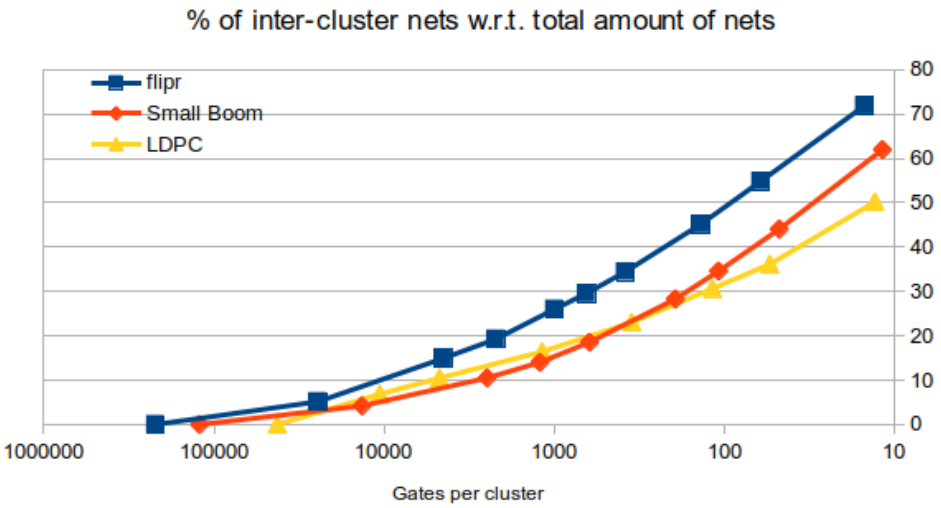

Figure 5.3: Percentage of inter-cluster nets *w.r.t.* the total amount of nets for the 7nm designs.

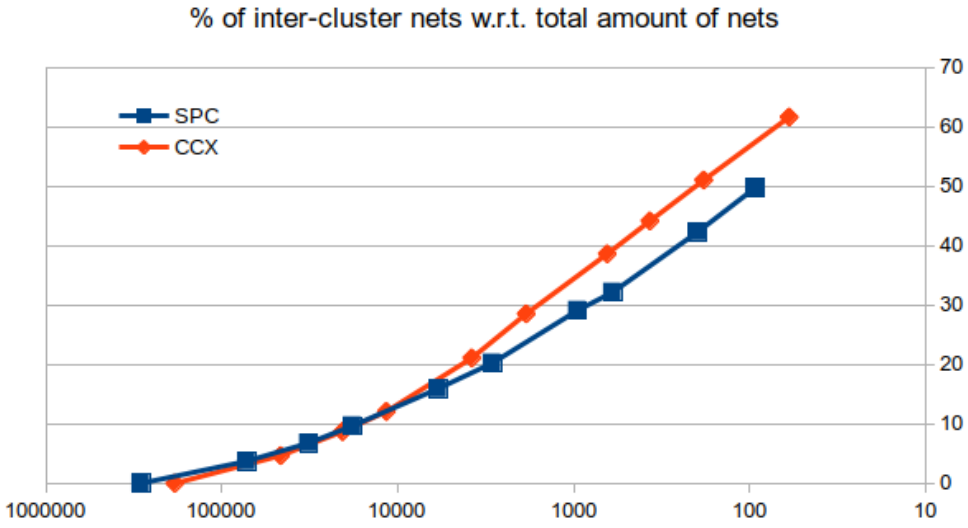

Figure 5.4: Percentage of inter-cluster nets *w.r.t.* the total amount of nets for the 45nm designs.

First, the figures 5.3 and 5.4 show the number of nets that span over several clusters. As it was expected, the proportion increases the finner the clustering gets. All five designs follow the same trend with some disparities when the clusters become really small, which is not surprising either. With very small clusters, the particularities of each design tend to take over the general trend.

From those results, we can say that the finner the grain, the more nets will need to be considered in the partitioning, but we have some margin. Indeed, most automatic clustering tools (either hierarchical, logical or geometrical) limit themselves from a few dozens up to a hundredish clusters. Based on table 5.2, for 100 nodes, we are well under 30% of total nets that will play

a role in the partitioning.

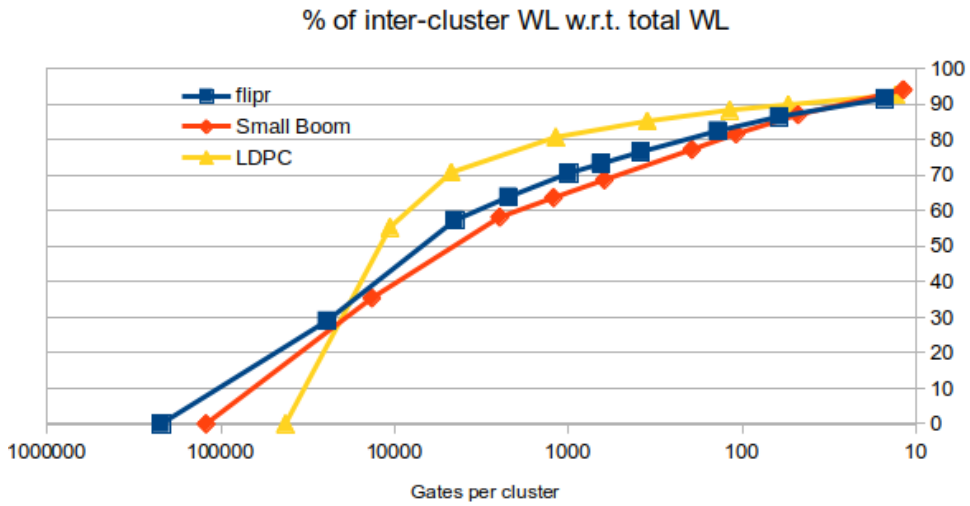

Figure 5.5: Percentage of inter-cluster WL *w.r.t.* the total WL for the 7nm designs.

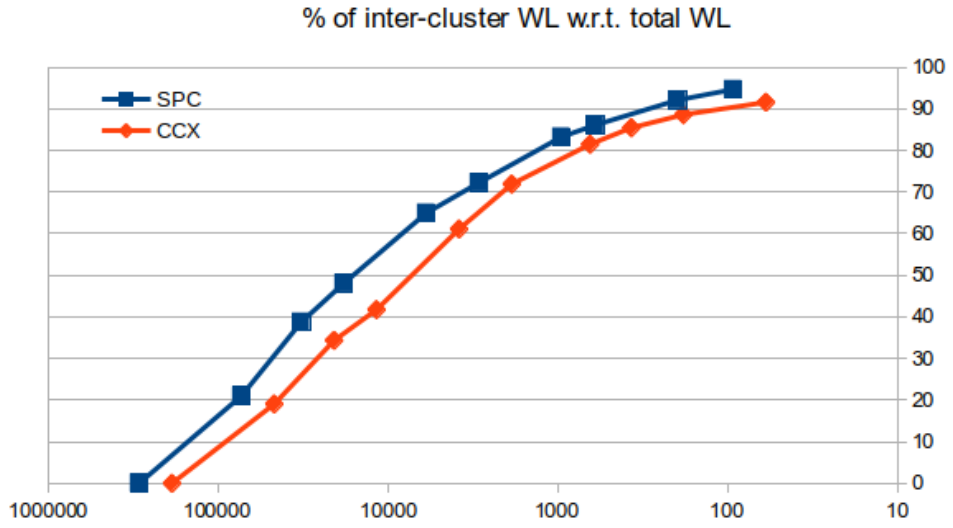

Figure 5.6: Percentage of inter-cluster WL *w.r.t.* the total WL for the 45nm designs.

Second, the figures 5.5 and 5.6 again show the same tendencies for both technology nodes and all designs (with LDPC deviating a little). If we consider the same situation as before, that is 100 nodes, between 60% and 80% of the total wire length is spanning several clusters and may be cut during the partitioning.

The objective of the clustering is to have as little nets to cut as possible, while still playing with as much wire length as possible. To find the sweet spot, both metrics need to be analyzed at the same time. When comparing the pairs of graphs for each technology node, a clustering embedding between 1000 and 5000 gates seems to be a nice match.

5.4 Intra-cluster connectivity

The intra-cluster connectivity counts the number of nets that are entirely contained inside a single cluster. This metric helps identifying the different zones of the architecture: highly or loosely connected.

For our intra-cluster study, we will focus on the LDPC architecture, as the other designs gave similar results.

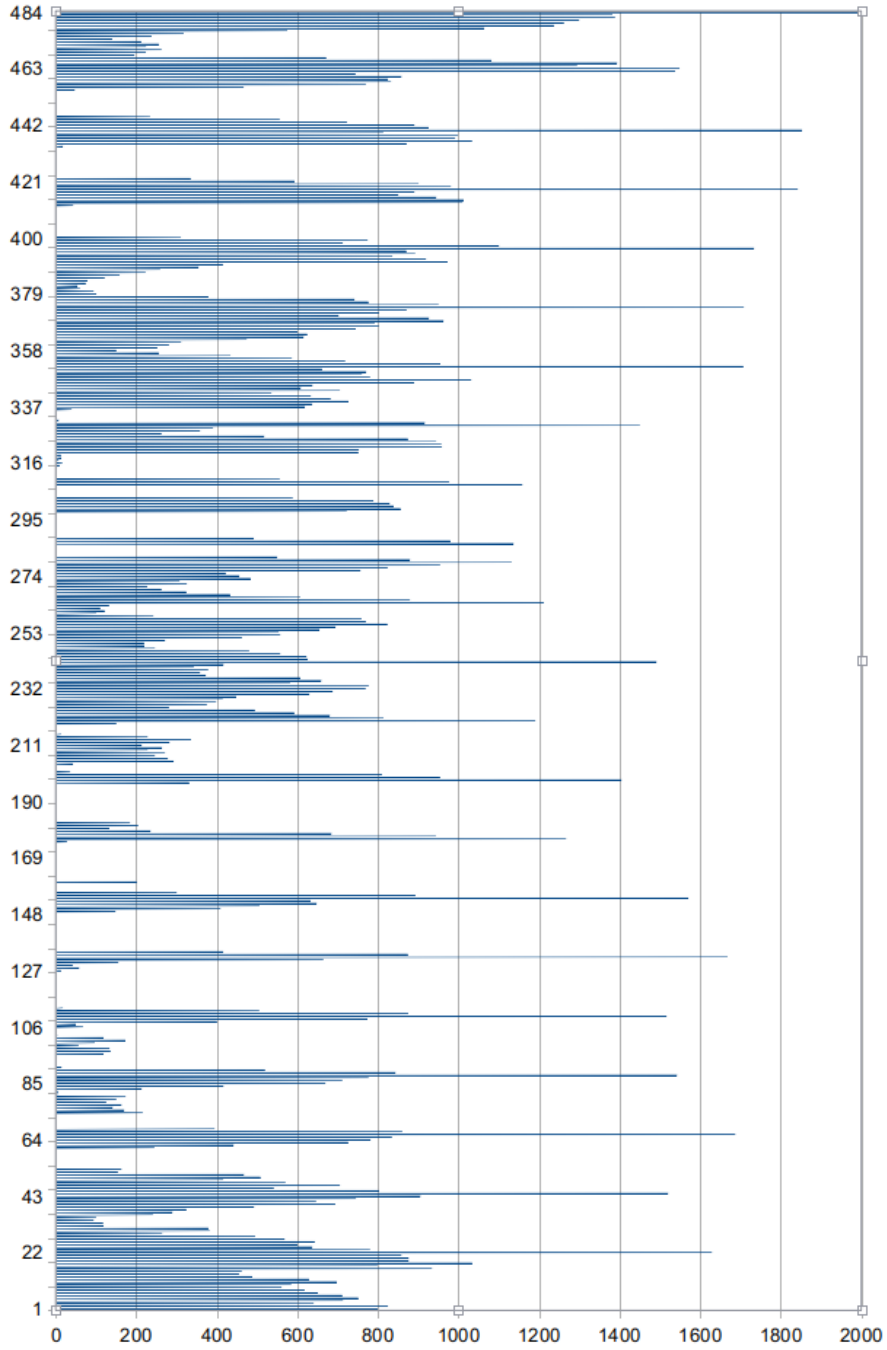

Figure 5.7: Intra-cluster connectivity for the LDPC architecture. The clustering applied is geometrical and each cluster is numbered from 1 in the lower left corner of the die to 484 in the top right corner and progressing on horizontal rows of 22 clusters.

A pattern emerges from the figure 5.7. We can see that the clusters on the edge of the die have

a lot of intra-connections. This can be explained by the pins and I/O being on the border as well. More interestingly, it shows us that a lot of clusters do not have any exclusively-intra nets. When our partitioning objective is to reduce the 3D nets, we should not increase the number of clusters at the borders as it would increase the inter-cluster connectivity and potentially the amount of 3D nets. On the other hand, the clusters without intra-nets could be even more reduced without creating new intercluster nets.

This metric could be used to move our clustering from naive to a bit more clever: by identifying the parts with higher intra-connectivity density, we could implement an adaptative grain coarsening, coarser at the border and finner in the inside.

Chapter 6

Future

Many questions motivate this work and many more arise. However, we need to focus on a subset of the problem pool.

First, the comparison between monolithic and stacking: which is the best choice? Toward which technology turn if we want efficient and profitable 3D ICs? We hope that our research into clustering grain and interconnectivity will help this problem moving forward.

Second, the lack of open source 3D toolchain. By developing each link in our chain as openly and flexible as possible, we try to work toward this objective.

Third, the limitation of layers. When talking about 3D integration today, we are still limited to two dies. In the future, we hope to lift this limitation and open the automated 3D integration to n layers.

This final chapter present some of the future directions our work will take.

6.1 Verilog partitioning

When partitioning an architecture, the output is a directive file telling the P&R tool on which die to place each cluster or standard cell. The tool then generates the appropriate signals for the dies to communicate together and a gate-level netlist for each die that will then be floorplanned, placed and routed as usual.

The next step in the tool independence is to develop the netlist generation ourselves. Doing so will require each cut net to be converted into an external signal on both ends of the cut and connecting it correctly in the netlist.

6.2 Accessibility/UI

Although the toolchain is currently working, it is quite cumbersome to use. All the configuration is set through a bunch of global variables and the lack of documentation is not helping.

In order for the toolchain to gain in quality, a proper user interface needs to be developed and some documentation to be written.

6.3 Clustering

As presented in section 2.2.5, the multilevel paradigm already clusters the vertices of the graph during the coarsening phase. The purpose of the naive clustering we are applying on the de-

sign, is (i) to reduce the order of the graph before feeding it to the partitioner, and (ii) maintain the integrity of logical blocks, hence easing the subsequent P&R of the partitioned dies. At the moment, we do not yet need a very efficient clustering for the current purpose. A future improvement could be to develop a more sophisticated algorithm that could replace the coarsening and uncoarsening of the multilevel paradigm, allowing us to fall back on a raw graph partitioning algorithm.

This would also be the occasion to test the validity of the discussion between gate- and block-level partitioning. With the rise of multilevel schemes, a gate-level graph would need to be coarsened before the actual partitioning can take place. That clustering would group the gates into pseudo-logical blocks (depending on the coarsening algorithm used) that could be close to the block-level graph. In that case, is there even a sense in differentiating the two approaches?

Furthermore, as discussed in section 5.3, the stake of clustering is finding the sweet spot between a low number of nets cut, and a high amount of wire length to play with and decrease. A way to achieve this may be to exploit the results of section 5.4 and develop an adaptative clustering method that would preserve the integrity of block with short length connections.

6.4 New metrics

Now that we are able to analyze the connectivity of clusters, it would be interesting to implement those analysis into the partitioner through a new set of weights. Here follows some weights worth investigating:

- The ratio of nets inside a hyperedge *w.r.t.* the total amount of nets.
- The ratio of inter- over intra-cluster nets.
- A linear ranking based on the wire length, with the highest weight granted to the longest net.

6.5 Multi-die partitioning

If two-way partitioning is pretty straightforward, stacking three or more dies becomes more difficult. How do we ensure the N+1 die can be connected with the N-1? How do we penalize the connection between distant layers in the hypergraph representation? Those are some questions that need answers if we want to enable multi-die integration in our workflow.

6.6 Extended support for partitioners

We intend to extend the suport of `PHONEY` to new partitioning libraries and improve their configurability through the script.

Bibliography

[1] Y. Akasaka and T. Nishimura. Concept and basic technologies for 3-D IC structure. *1986 International Electron Devices Meeting*, pages 488–491, 1986. ISSN 01631918. doi: 10.1109/IEDM.1986.191227. URL `http://ieeexplore.ieee.org/document/1486485/`.

[2] K. Athikulwongse, M. Ekpanyapong, and S. K. Lim. Exploiting die-to-die thermal coupling in 3-D IC placement. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 22(10):2145–2155, 2014. ISSN 10638210. doi: 10.1109/TVLSI.2013.2285593.

[3] C. Aykanat, B. B. Cambazoglu, and B. Uçar. Multi-level direct K-way hypergraph partitioning with multiple constraints and fixed vertices. *Journal of Parallel and Distributed Computing*, 68(5):609–625, 2008. ISSN 07437315. doi: 10.1016/j.jpdc.2007.09.006.

[4] P. Batude, C. Fenouillet-Beranger, L. Pasini, V. Lu, F. Deprat, L. Brunet, B. Sklenard, F. Piegas-Luce, M. Casse, B. Mathieu, O. Billoint, G. Cibrario, O. Turkyilmaz, H. Sarhan, S. Thuries, L. Hutin, S. Sollier, J. Widiez, L. Hortemel, C. Tabone, M. P. Samson, B. Previtali, N. Rambal, F. Ponthenier, J. Mazurier, R. Beneyton, M. Bidaud, E. Josse, E. Petitprez, O. Rozeau, M. Rivoire, C. Euvard-Colnat, A. Seignard, F. Fournel, L. Benaissa, P. Coudrain, P. Leduc, J. M. Hartmann, P. Besson, S. Kerdiles, C. Bout, F. Nemouchi, A. Royer, C. Agraffeil, G. Ghibaudo, T. Signamarcheix, M. Haond, F. Clermidy, O. Faynot, and M. Vinet. 3DVLSI with CoolCube process: An alternative path to scaling. *Digest of Technical Papers - Symposium on VLSI Technology*, 2015-August:T48–T49, 2015. ISSN 07431562. doi: 10.1109/VLSIT.2015.7223698.

[5] L. Brunet, P. Batude, C. Fenouillet-Beranger, P. Besombes, L. Hortemel, F. Ponthenier, B. Previtali, C. Tabone, A. Royer, C. Agraffeil, C. Euvrard-Colnat, A. Seignard, C. Morales, F. Fournel, L. Benaissa, T. Signamarcheix, P. Besson, M. Jourdan, R. Kachtouli, V. Benevent, J. M. Hartmann, C. Comboroure, N. Allouti, N. Posseme, C. Vizioz, C. Arvet, S. Barnola, S. Kerdiles, L. Baud, L. Pasini, C. M. V. Lu, F. Deprat, A. Toffoli, G. Romano, C. Guedj, V. Delaye, F. Boeuf, O. Faynot, and M. Vinet. First demonstration of a CMOS over CMOS 3D VLSI CoolCube integration on 300mm wafers. *Digest of Technical Papers - Symposium on VLSI Technology*, 2016-Septe:11–12, 2016. ISSN 07431562. doi: 10.1109/VLSIT.2016.7573428.

[6] A. E. Caldwell, A. B. Kahng, and I. L. Markov. Improved algorithms for hypergraph bipartitioning. *Proceedings of the 2000 Asia and ...*, pages 661–666, 2000. doi: 10.1109/ASPDAC.2000.835182. URL `http://dl.acm.org/citation.cfm?id=368864`.

[7] Ü. Çatalyürek and C. Aykanat. PaToH (Partitioning Tool for Hypergraphs), 2011. URL `http://link.springer.com/content/pdf/10.1007/978-0-387-09766-4{_}93.pdf`.

[8] K. Chang, S. Sinha, B. Cline, R. Southerland, M. Doherty, G. Yeric, and S. K. Lim. Cascade2D: A design-aware partitioning approach to monolithic 3D IC with 2D commercial tools. *Proceedings of the 35th International Conference on Computer-Aided Design - ICCAD '16*, pages 1–8, 2016. ISSN 10923152. doi: 10.1145/2966986.2967013. URL `http://dl.acm.org/citation.cfm?doid=2966986.2967013`.

[9] F. Clermidy, O. Billoint, H. Sarhan, and S. Thuries. Technology scaling: The CoolCube paradigm. *2015 IEEE SOI-3D-Subthreshold Microelectronics Technology Unified Conference, S3S 2015*, pages 2–5, 2015. doi: 10.1109/S3S.2015.7333504.

[10] J. P. Colinge and E. Demoulin. ST-CMOS -stacked Transistors CMOS): A Double-Poly-NMOS-Compatible CMOS Technology. In *IEDM 81*, pages 557–561, 1981.

[11] J. Cong and J. Wei. A thermal-driven floorplanning algorithm for 3D ICs. *Computer Aided Design*, pages 306–313, 2004. ISSN 1092-3152. doi: 10.1109/ICCAD.2004.1382591. URL `http://ieeexplore.ieee.org/xpls/abs{_}all.jsp?arnumber=1382591`.

[12] J. Cong, A. Jagannathan, Y. Ma, G. Reinman, J. Wei, and Y. Zhang. An automated design flow for 3D microarchitecture evaluation. *Proceedings of the 2006 Asia and South Pacific Design Automation Conference*, pages 384–389, 2006. doi: http://dx.doi.org/10.1145/1118299.1118395. URL `http://dx.doi.org/10.1145/1118299.1118395`.

[13] J. Cong, G. Luo, J. Wei, and Y. Zhang. Thermal-aware 3D IC placement via transformation. *Proceedings of the Asia and South Pacific Design Automation Conference, ASP-DAC*, pages 780–785, 2007. doi: 10.1109/ASPDAC.2007.358084.

[14] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms*. MIT Press, third edition, 2009. ISBN 978-0-262-03384-8.

[15] K. D. Devine, E. G. Boman, R. T. Heaphy, R. H. Bisseling, and U. V. Catalyurek. Parallel hypergraph partitioning for scientific computing. In *20th International Parallel and Distributed Processing Symposium, IPDPS 2006*, volume 2006, 2006. ISBN 1424400546. doi: 10.1109/IPDPS.2006.1639359.

[16] S. Dutt. New Faster Kernighan-Lin-Type Graph-Partitioning Algorithms. In *Computer-Aided Design, 1993. ICCAD-93. Digest of Technical Papers., 1993 IEEE/ACM International Conference on*, 1993. doi: 10.1109/ICCAD.1993.580083.

[17] C. M. Fiduccia and R. M. Mattheyses. A Linear-Time Heuristic for Improving Network Partitions. *Design Automation, 1982. 19th Conference on*, pages 175–181, 1982. ISSN 0146-7123. doi: 10.1109/DAC.1982.1585498.

[18] M. R. Garey, D. S. Johnson, and L. Stockmeyer. Some simplified NP-complete graph problems. *Theoretical Computer Science*, 1(3):237–267, 1976. ISSN 03043975. doi: 10.1016/0304-3975(76)90059-1.

[19] W. Geis, D. C. Flanders, D. A. Antoniadis, and I. Smith. Crystalline silicon on insulators by graphoepitaxy*. In *Electron Devices Meeting, 1979 International*, pages 210–212, 1979.

[20] G. T. Goeloe, E. W. Maby, D. J. Silversmith, R. W. Mountain, and D. A. Antoniadis. Vertical single-gate CMOS inverters on laser-processed multilayer substrates. In *Electron Devices Meeting, 1981 International*, pages 554–556, 1981.

[21] L. Hagen and A. B. Kahng. New Spectral Methods for Ratio Cut Partitioning and Clustering. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 11 (9):1074–1085, 1992. ISSN 19374151. doi: 10.1109/43.159993.

[22] B. Hendrickson and R. Leland. The Chaco user's guide, version 2.0. Technical Report SAND94–2692, 1994.

[23] E. Ihler, D. Wagner, and F. Wagner. Modeling Hypergraphs by Graphs with the same Mincut Properties. *Information Processing Letters*, 45:171—-175, 1993. ISSN 0002-9572. doi: 10.1016/0020-0190(93)90115-P.

[24] P. Jain, T.-h. Kim, J. Keane, and C. H. Kim. A multi-story power delivery technique for 3D integrated circuits. *Proceeding of the thirteenth international symposium on Low power electronics and design - ISLPED '08*, page 57, 2008. ISSN 9781605581095. doi: 10.1145/1393921.1393940. URL `http://portal.acm.org/citation.cfm?doid=1393921.1393940`.

[25] D. Johnson, C. Aragon, L. McGeoch, and C. Schevon. Optimization by Simulated Annealing: An Experimental Evaluation; Part I, Graph Partitioning. *Operations Research*, 37(6): 865–893, 1989.

[26] M. Jung, D. Z. Pan, and S. K. Lim. Chip/package co-analysis of thermo-mechanical stress and reliability in TSV-based 3D ICs. *Proceedings of the 49th Annual Design Automation Conference on - DAC '12*, page 317, 2012. ISSN 0738-100X. doi: 10.1145/2228360.2228419. URL `http://dl.acm.org/citation.cfm?doid=2228360.2228419`.

[27] A. B. Kahng, J. Lienig, I. L. Markov, and J. Hu. *VLSI physical design: From graph partitioning to timing closure*, volume 25. Springer Netherlands, 2011. ISBN 9789048195909. doi: 10.1007/978-90-481-9591-6.

[28] G. Karypis and V. Kumar. Analysis of Multilevel Graph Partitioning. *Supercomputing*, pages 1–19, 1995. ISSN 10639535. doi: 10.1145/224170.224229.

[29] G. Karypis and V. Kumar. Multilevel Graph Partitioning Schemes. *[ICPP'95] International Conference on Parallel Processing*, 3:113–122, 1995. ISSN 01903918. doi: 10.1145/224170.224229. URL `http://static.msi.umn.edu/rreports/1995/113.pdf`.

[30] G. Karypis, R. Aggarwal, V. Kumar, and S. Shekhar. Multilevel hypergraph partitioning: Applications in VLSI domain. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 7(1):69–79, 1999. ISSN 10638210. doi: 10.1109/92.748202. URL `http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=748202`.

[31] S. Kawamura, N. Sasaki, T. Iwai, M. Nakano, and A. M. Takagi. Three-dimensional CMOS IC's fabricated by using beam recrystallization. *Ieee Electron Device Letters*, 4(10):366–368, 1983.

[32] B. W. Kernighan and S. Lin. An Efficient Heuristic Procedure for Partitioning Graphs. *Bell System Technical Journal*, 49(2):291–307, 1970. ISSN 15387305. doi: 10.1002/j.1538-7305.1970.tb01770.x.

[33] J.-M. Koo, S. Im, L. Jiang, and K. E. Goodson. Integrated Microchannel Cooling for Three-Dimensional Electronic Circuit Architectures. *Journal of Heat Transfer*, 127(1):

49, 2005. ISSN 00221481. doi: 10.1115/1.1839582. URL http://heattransfer.asmedigitalcollection.asme.org/article.aspx?articleid=1447600.

[34] T. Kunio, K. Oyama, Y. Hayashi, and M. Morimoto. Three dimensional ICs, having four stacked active device layers. *International Technical Digest on Electron Devices Meeting*, pages 837–840, 1989. ISSN 01631918. doi: 10.1109/IEDM.1989.74183. URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=74183{%}5Cnhttp://ieeexplore.ieee.org/document/74183/.

[35] Y.-J. Lee, D. Limbrick, and S. K. Lim. Power benefit study for ultra-high density transistor-level monolithic 3D ICs. *Proceedings of the Design Automation Conference*, pages 1–6, 2013. ISSN 0738100X. doi: 10.1145/2463209.2488863. URL http://dl.acm.org/citation.cfm?id=2463209.2488863.

[36] S. C. Lin and K. Banerjee. Cool chips: Opportunities and implications for power and thermal management. *IEEE Transactions on Electron Devices*, 55(1):245–255, 2008. ISSN 00189383. doi: 10.1109/TED.2007.911763.

[37] F. Lotfifar and M. Johnson. A multi-level hypergraph partitioning algorithm using rough set clustering. In J. L. Träff, S. Hunold, and F. Versaci, editors, *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 9233, pages 159–170. Springer Berlin Heidelberg, 2015. ISBN 9783662480953. doi: 10.1007/978-3-662-48096-0_13.

[38] T. Lu, C. Serafy, Z. Yang, S. Samal, S. K. Lim, and A. Srivastava. TSV-based 3D ICs: Design Methods and Tools. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 0070(c):1–1, 2017. ISSN 0278-0070. doi: 10.1109/TCAD.2017.2666604. URL http://ieeexplore.ieee.org/document/7849155/.

[39] J. Michailos, P. Coudrain, A. Farcy, N. Hotellier, S. Cheramy, S. Lhostis, E. Deloffre, Y. Sanchez, A. Jouve, F. Guyader, E. Saugier, V. Fiori, P. Vivet, M. Vinet, C. Fenouillet-Beranger, F. Casset, P. Batude, F. Breuf, Y. Henrion, B. Vianne, L. M. Collin, J. P. Colonna, L. Benaissa, L. Brunet, R. Prieto, R. Velard, and F. Ponthenier. New challenges and opportunities for 3D integrations. *Technical Digest - International Electron Devices Meeting, IEDM*, 2016-February:8.5.1–8.5.4, 2016. ISSN 01631918. doi: 10.1109/IEDM.2015.7409655.

[40] J. Minz, X. Zhao, and S. K. Lim. Buffered clock tree synthesis for 3D ICs under thermal variations. *Proceedings of the Asia and South Pacific Design Automation Conference, ASP-DAC*, (i):504–509, 2008. doi: 10.1109/ASPDAC.2008.4484003.

[41] B. Monien, R. Preis, and R. Diekmann. Quality Matching And Local Improvement For Multilevel Graph-partitioning. *Parallel Computing*, 26(12):1609–1634, 2000. ISSN 01678191.

[42] M. Nakano. 3-D SOI/CMOS. In *IEDM 84*, pages 792–795, 1984.

[43] G. W. Neudeck, S. Pae, J. P. Denton, and T.-c. Su. Multiple layers of silicon-on-insulator for nanostructure devices. *Journal of Vacuum Science & Technology B: Microelectronics and Nanometer Structures*, 17(February):994, 1999. ISSN 0734211X. doi: 10.1116/1.590682.

[44] T. Nishimura, Y. Inoue, K. Sugahara, S. Kusunoki, T. Kumamoto, S. Nakagawa, M. Nakaya, Y. Horiba, and Y. Akasaka. Three dimensional IC for high performance image signal processor. In *Electron Devices Meeting, 1987 International*, pages 111–114, 1987. ISBN 2515518710.

[45] S. Panth, K. Samadi, Y. Du, and S. K. Lim. High-density integration of functional modules using monolithic 3D-IC technology. *Proceedings of the Asia and South Pacific Design Automation Conference, ASP-DAC*, pages 681–686, 2013. ISSN 2153-6961. doi: 10.1109/ASPDAC.2013.6509679.

[46] S. Panth, K. Samadi, Y. Du, and S. K. Lim. Power-Performance Study of Block-Level Monolithic 3D-ICs Considering Inter-Tier Performance Variations. *DAC, Proceedings of the Annual Design Automation Conference on Design Automation Conference*, pages 1–6, 2014. ISSN 0738100X. doi: 10.1145/2593069.2593188. URL `http://dl.acm.org/citation.cfm?doid=2593069.2593188`.

[47] S. a. Panth, K. Samadi, Y. Du, and S. K. Lim. Design and CAD methodologies for low power gate-level monolithic 3D ICs. *Proceedings of the 2014 international symposium on Low power electronics and design - ISLPED '14*, 1:171–176, 2014. ISSN 15334678. doi: 10.1145/2627369.2627642. URL `http://dl.acm.org/citation.cfm?doid=2627369.2627642`.

[48] S. Pantht, K. Samadp, Y. Du, and S. K. Limt. Tier-Partitioning for Power Delivery vs Cooling Tradeoff in 3D VLSI for Mobile Applications. *ACM/EDAC/IEEE Design Automation Conference (DAC)*, pages 1–6, 2015. ISSN 0738100X. doi: 10.1145/2744769.2744917. URL `http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=7167276`.

[49] Y. Peng, D. Petranovic, K. Samadi, P. Kamal, Y. Du, and S. K. Lim. Inter-die Coupling Extraction and Physical Design Optimization for Face-to-Face 3D ICs. *IEEE Transactions on Nanotechnology*, (c):1–1, 2017. ISSN 1536-125X. doi: 10.1109/TNANO.2017.2735361. URL `http://ieeexplore.ieee.org/document/8000352/`.

[50] M. Pirlot. General local search methods. *European Journal of Operational Research*, 92 (3):493–511, 1996. ISSN 03772217. doi: 10.1016/0377-2217(96)00007-0.

[51] S. K. Ryu, K. H. Lu, X. Zhang, J. H. Im, P. S. Ho, and R. Huang. Impact of near-surface thermal stresses on interfacial reliability of through-silicon vias for 3-D Interconnects. *IEEE Transactions on Device and Materials Reliability*, 11(1):35–43, 2011. ISSN 15304388. doi: 10.1109/TDMR.2010.2068572.

[52] S. K. Samal, K. Samadi, P. Kamal, Y. Du, and S. K. Lim. Full chip impact study of power delivery network designs in monolithic 3D ICs. *2014 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 0070(c):565–572, 2014. ISSN 0278-0070. doi: 10.1109/ICCAD.2014.7001406. URL `http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=7001406`.

[53] K. Saraswat, S. Souri, V. Subramanian, A. Joshi, and A. Wang. Novel 3-D structures. *1999 IEEE International SOI Conference. Proceedings (Cat. No.99CH36345)*, pages 54–55, 1999. doi: 10.1109/SOI.1999.819855. URL `http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=819855`.

[54] C. Serafy, B. Shi, A. Srivastava, and D. Yeung. High performance 3D stacked DRAM processor architectures with micro-fluidic cooling. *2013 IEEE International 3D Systems Integration Conference, 3DIC 2013*, 2013. doi: 10.1109/3DIC.2013.6702353.

[55] C. Serafy, A. Bar-Cohen, A. Srivastava, and D. Yeung. Unlocking the true potential of 3-D CPUs with microfluidic cooling. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 24(4):1515–1523, 2016. ISSN 10638210. doi: 10.1109/TVLSI.2015.2450192.

[56] P. Singer. Exponentially rising costs will bring changes, 2014. URL `http://electroiq.com/petes-posts/2015/01/26/exponentially-rising-costs-will-bring-changes/`.

[57] S. Souri, K. Banerjee, A. Mehrotra, and K. Saraswat. Multiple Si layer ICs: motivation, performance analysis, and design implications. *Design Automation Conference, 2000. Proceedings 2000*, pages 213–220, 2000. doi: 10.1109/DAC.2000.855306.

[58] S. Strickland, E. Ergin, D. R. Kaeli, and P. Zavracky. VLSI design in the 3rd dimension. *Integration, the VLSI Journal*, 25(1):1–16, 1998. ISSN 01679260. doi: 10.1016/S0167-9260(98)00006-6.

[59] V. Subramanian and K. C. Saraswat. High-performance germanium-seeded laterally crystallized TFT's for vertical device integration. *IEEE Transactions on Electron Devices*, 45 (9):1934–1939, 1998. ISSN 00189383. doi: 10.1109/16.711358.

[60] K. Sugahara, T. Nishimura, S. Kusunoki, Y. Akasaka, and H. Nakata. SOI/SOI/Bulk-Si Triple-Level Structure for Three-Dimensional Devices. *IEEE Electron Device Letters*, 7 (3):193–195, 1986. ISSN 15580563. doi: 10.1109/EDL.1986.26341.

[61] C. S. Tan, R. J. Gutmann, and L. R. Reif. *Wafer Level 3-D ICs Process Technology*. Springer US, 2008. ISBN 9780387765327. doi: 10.1007/978-0-387-76534-1.

[62] S. A. Terui, S. Ogawa, M. Yoneda, N. Yoshii, and Y. Terui. Multilayer CMOS device fabricated on laser recrystallized silicon islands. In *Electron Devices Meeting, 1983 International*, pages 352–355, 1983.

[63] A. Trifunovi. *Department of Computing Parallel Algorithms for Hypergraph Partitioning Aleksandar Trifunovi ´.* PhD thesis, University of London Imperial College of Science, Technology and Medicine, 2006.

[64] A. Trifunovic and W. J. W. Knottenbelt. Parkway 2.0: A parallel multilevel hypergraph partitioning tool. *Computer and Information Sciences - ISCIS 2004*, pages 789–800, 2004. ISSN 03029743. doi: 10.1007/978-3-540-30182-0_79. URL `http://link.springer.com/chapter/10.1007/978-3-540-30182-0{_}79`.

[65] C. Tsourakakis, C. Gkantsidis, B. Radunovic, and M. Vojnovic. Fennel: Streaming graph partitioning for massive scale graphs. *Proceedings of the 7th ACM international conference on Web search and data mining - WSDM '14*, pages 333–342, 2014. doi: 10.1145/2556195.2556213. URL `http://dl.acm.org/citation.cfm?doid=2556195.2556213`.

[66] S. R. Vempati, N. Su, C. H. Khong, Y. Y. Lim, K. Vaidyanathan, J. H. Lau, B. P. Liew, K. Y. Au, S. Tanary, A. Fenner, R. Erich, and J. Milla. Development of 3-D silicon die stacked

package using flip chip technology with micro bump Interconnects. *Proceedings - Electronic Components and Technology Conference*, pages 980–987, 2009. ISSN 05695503. doi: 10.1109/ECTC.2009.5074132.

[67] M. Vinet, P. Batude, C. Fenouillet-Beranger, L. Brunet, V. Mazzochi, C. M. V. Lu, F. Deprat, J. Micout, B. Previtali, P. Besombes, N. Rambal, F. Andrieu, O. Billoint, M. Brocard, S. Thuries, G. Berhault, C. L. Dos Santos, G. Cibrario, F. Clermidy, D. Gitlin, and O. Faynot. Opportunities brought by sequential 3D CoolCube integration. *European Solid-State Device Research Conference*, 2016-Octob:226–229, 2016. ISSN 19308876. doi: 10.1109/ESSDERC.2016.7599627.

[68] L. Wang, Y. Xiao, B. Shao, and H. Wang. How to partition a billion-node graph. Technical Report MSR-TR-2013-102, feb 2014. URL `http://research.microsoft.com/apps/pubs/default.aspx?id=183714`.

[69] R. Wittmann. *Miniaturization Problems in CMOS Technology: Investigation of Doping Profiles and Reliability*. PhD thesis, Technischen Universität Wien, 2007. URL `http://www.iue.tuwien.ac.at/phd/wittmann/`.

[70] Q. Xu, L. Jiang, H. Li, and B. Eklow. Yield enhancement for 3D-stacked ICs: Recent advances and challenges. *Proceedings of the Asia and South Pacific Design Automation Conference, ASP-DAC*, pages 731–737, 2012. doi: 10.1109/ASPDAC.2012.6165052. URL `http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6165052`.

[71] G. Yeap. Smart mobile SoCs driving the semiconductor industry: Technology trend, challenges and opportunities. *Technical Digest - International Electron Devices Meeting, IEDM*, pages 16–23, 2013. ISSN 01631918. doi: 10.1109/IEDM.2013.6724540.

[72] T. Zhang, Y. Zhan, and S. S. Sapatnekar. Temperature-aware routing in 3D ICs. *Proceedings of the 2006 Asia and South Pacific Design Automation Conference*, pages 309–314, 2006. doi: http://dx.doi.org/10.1145/1118299.1118377. URL `http://dx.doi.org/10.1145/1118299.1118377`.