1

Problem definition and 3D-stacked integrated circuit model

In the previous chapters, we have presented a review of the literature about the field of microelectronics design. We have highlighted some limitations of the current tools that already occur for current ICs. In this chapter, we will define the problem we tackle and show how we model it in order to propose improvements to design flows.

1.1 Problem definition

As stated in Chapter **??**, the limitations of the current design flows can be summarized in three points:

— Limitation of the design space exploration
— Unicriterion optimization
— No 3D-SIC specific tools

In order to address these limitations, we propose in this thesis a methodology based on multi-objective/criteria tools and taking into account 3D-SIC specificities for an exploration of the design space.

While this methodology could be applied at different levels in a design flow, we have focused our development in the logical design step and more specifically in the virtual prototyping flow with performance assessments.

1.1.1 Design an IC

In order to meet the specifications, a design has to first make choice at a physical level:

— Targeted architecture, e.g. ASIC, FPGA
— Number of functional units
— Number of memories and their size
— The general layout
— ...

Since the 3D-SICs are based on conventional circuits, the options and degrees of freedom coming from 2D-ICs are still present:

— Process technology, e.g. 180 nm to 22 nm CMOS
— Memories technology, e.g. SRAM, DRAM, FLASH
— Communication infrastructure, e.g. bus, Network-on-Chip
— ...

In addition to those options and degrees of freedom coming from 2D-ICs, there are also numerous 3D-SIC's parameters:

— Number of tiers to use
— Place and route of the functional units between the tiers
— Technology to use per tiers (heterogeneity)
— Interconnection and geometry between tiers
— 3D-SIC integration technology
— 3D-SIC assembly technology
— ...

The above mentioned parameters illustrate the numerous possibilities for designing a circuit and how the design space for 2D-ICs becomes much bigger when considering 3D-SICs. The main issue is therefore to choose the most efficient combination among all those options. This can thus be compared to a combinatorial optimization problem which makes the use of metaheuristics more than justified as they are commonly-used tools for such problems. Also, given the multi-criteria nature of designing 3D-SIC, this optimization will have to take into account all the criteria simultaneously. In the next section will define the criteria that a designer can consider.

1.2 Model and criteria definition

Typically, the criteria that have to be optimized simultaneously can be the performance, the power consumption, the cost, the package size, the heat dissipation, etc. In this model, we will define six criteria which are among the most important parameters while designing a circuit [1]. Among the six presented criteria, the first five are implemented and the latter one is discussed:

1. *The interconnection global length*: this parameter can reflect the global performance of a system. The objective is to minimize it in order to have, for instance, a short delay and low power consumption. It will be calculated using the Manhattan distance:

$$d_{i,j} = |x_i - x_j| + |y_i - y_j| \tag{1.1}$$

where (x_n, y_n) is the geometrical coordinates of the n^{th} block. As a first approximation, the center point of each block will be selected as reference coordinate. Also, since it is more interesting to place close to each other two blocks that require a large bandwidth (BW) to communicate, we will balance the values as follows:

$$d'_{i,j} = \frac{d_{i,j}}{BW_{i,j}} \tag{1.2}$$

So the global interconnection length D will be the sum of $d'_{i,j}$ for all communicating blocks:

$$D = \sum d'_{i,j} \tag{1.3}$$

2. *The cost*: an economical factor is obviously an important criteria for a design. This criteria has been estimated with the aid of an expert in 3D-SIC manufacturing. While a circuit can be more efficient with many layers, it will also be more expensive. This criteria has to be minimized. Due to the confidential nature of the cost of a 3D-SIC, we will consider a simplified model where the cost is proportional to the area and increasing exponentially with the number of tiers:

$$cost = a(tech).S + b(tech)^{layer\ number} \tag{1.4}$$

where $a(tech)$ and $b(tech)$ are coefficient depending on the technology assigned. Let us note that this criterion includes both discrete and continuous variables.

3. *The package volume*: this can be an important criteria when designing embedded circuits. The package volume is calculated as follows:

$$volume = largest\ layer\ size\ *\ stack\ thickness \tag{1.5}$$

A large appoximation of 200 μm will be made for the thickness of one tiers. Let us note that this criterion includes both discrete and continuous variables.

4. *The clock tree position*: in this model, we consider a synchronous system so the objective is to minimize the distance between each block and the clock tree in order to have a high frequency. We choose arbitrarily to approximate the reference point as a fixed point located at the upper left corner of the middle tier of the 3D-SIC.

5. *The thermal dissipation*: thermal dissipation is one of the major issues when designing 3D-SICs. It can be more appropriate to place two blocks underneath each other in successive tiers but a high heat dissipation may happen in intensive computational process. This criterion is a research topic on its own [2, 3]. Here we will use a simplified evaluation model with finite elements. This model will consider that the dissipated power, intra- or inter-tiers, is inversely proportional to the distance to the heat source:

$$P_{diss} = \sum_i \frac{1}{R_{th,i} r} \tag{1.6}$$

where r is the distance to the heat source and $R_{th,i}$ the thermal resistance depending on whether the dissipation is intra- or inter- tiers. This criteria is still on early development stage and we can generate thermal maps of a floorplan as shown in Fig. 1.1 but this is currently based on finite elements [4] which require quite a long computational time even for a simplified thermal model. This criterion in its current development stage is difficult to integrate to the exploration process, due to the computation time of finite elements methods. In the current work, we will simply compute the peak power of a circuit which can be done more quickly.

6. *The power consumption*: the objective is to minimize the power consumption which can be a crucial criteria for embedded systems. Generally, the power consumption can be defined as the sum of a static (P_{stat}) (given data from the component datasheet) and a dynamic (P_{dyn}) consumption:

$$P = P_{stat} + P_{dyn} \tag{1.7}$$

$$P_{stat} = [given\ data] \tag{1.8}$$

$$P_{dyn} = \alpha.C.V_{dd}^2.f.[tech] \tag{1.9}$$

where α represents the toggle rate, C the capacitance, V_{dd} the voltage, f the frequency and $[tech]$ a rectification factor due to the technology assigned.

At first, we will focus on the three first criteria in order to be able to have a visualization of the design space. We will also arbitrarily introduce some limitations in term of degrees of freedom to analyse what happens if we release a degree of freedom. This will be done while considering the three same criteria, in order to keep a visualization of how the flexibility of MOO will improve the information and the results. Then we will analyse our methodology with the five first criteria that have been presented. As stated, the sixth criterion is currently unused. Actually, it has been simulated and tested but we need data from the manufacturers which are not easily available.

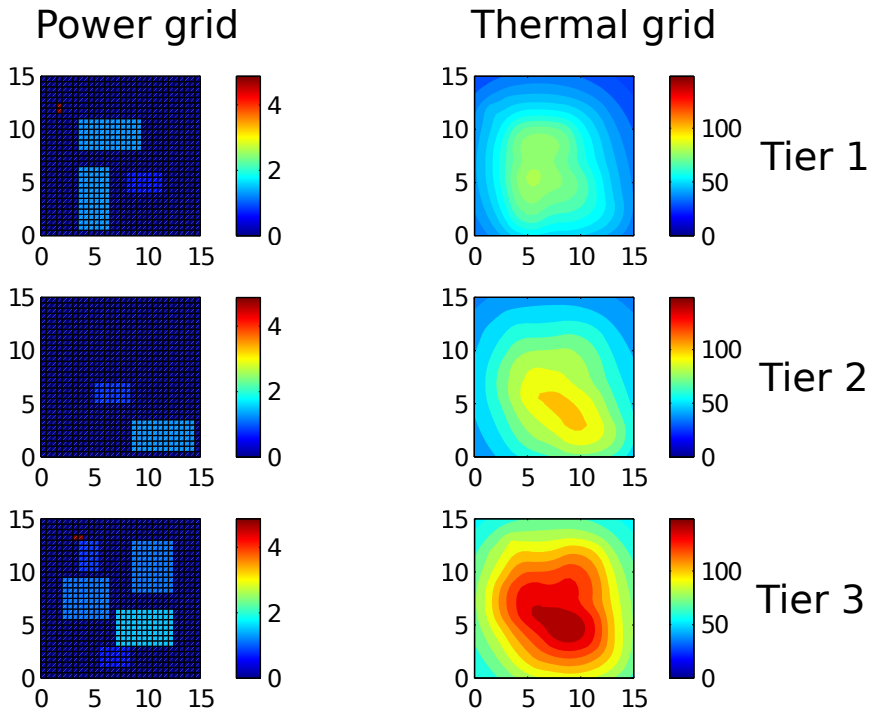

Figure 1.1: Power grid and thermal map of a floorplan (3 tiers)

1.3 Design methodology

In summary, the problem we are facing is to place several blocks that have to be assigned in many tiers while considering multiple conflicting criteria. Now that our model has been defined, we will present a proposition of a new design methodology based on multi-objective optimization.

As explained previously, designing ICs implies numerous choices. At the moment, with this growing complexity, the current design flows can show their limits. For instance, most of the time, the designers will be likely to freeze a certain amount of choices on basis of their experience, and then begin the optimization process with the remaining parameters. This will therefore limit the exploration of the design space and good solutions may be ignored. In addition, the fixed choices can be questionable since they are based on the designer's experience though they could also be based on more objective facts.

In order to enable an efficient design space exploration, we propose a method in four steps based on MCDA which is illustrated in Fig. 1.2. The implementation will be briefly presented in the next section.

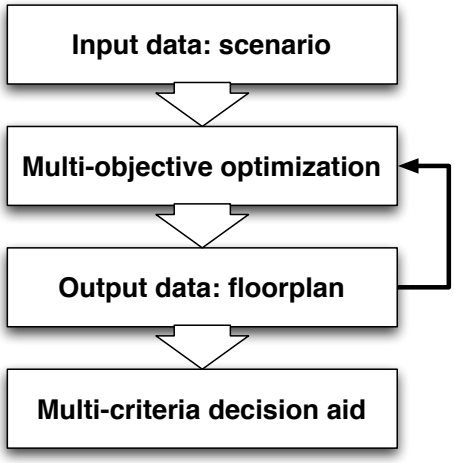

Figure 1.2: MCDA-based design methodology

For the problem we consider, the input data will contain the information about the scenario:

— Type and number of blocks: computational units, memories, etc.
— Size of the blocks: inherent to the block.
— Minimum aspect ratio: we consider a degree of freedom where a block can have its dimensions varying within an aspect ratio range. This means that a block does not have to be square, as shown in Fig. 1.3. This parameter can influence the delay in a block.
— Size variability of a block: we add this degree of freedom considering that the specified size of a block can be fixed by the designer but this fixed size can restrict the design space exploration. The variability of a block's size can have effect on the performance and the global footprint.

In addition, the bandwidth requirements are needed as they will indicate which are the important interconnections and prevent two blocks that require a large bandwidth from being too far from each other. The available manufacturing technologies are also useful to enable the design of heterogeneous systems.

The combination of all the parameters described in the model are the possible alternatives for a 3D-SIC design and will provide output data after design space exploration. For a floorplanning problem the required output data are generally the geometrical layout of the circuit [1]:

— The geometrical coordinates for each block and the assigned layer.
— The size of each block (if it can vary from the specified size).
— The aspect ratio for each block.
— The technology assigned to each tier: this will reduce the size of each block.

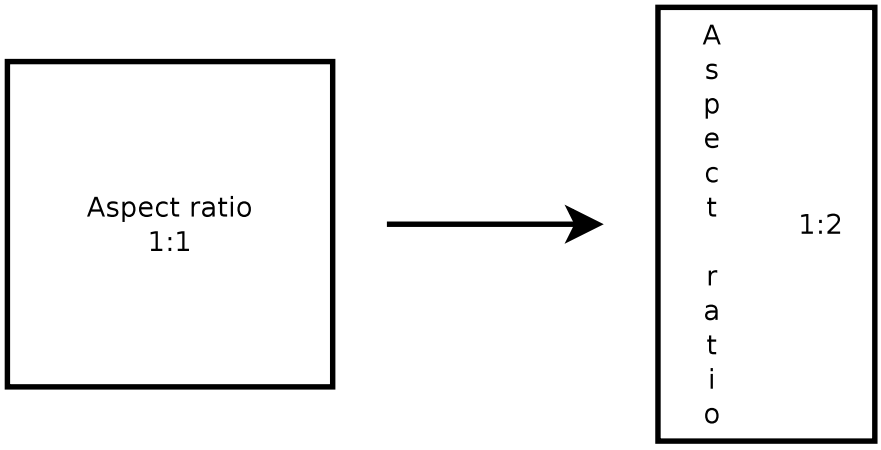

Figure 1.3: Example of aspect ratio degree of freedom

The size of a block will define the number of transistors inside using a given technology, for example 180 nm. For a constant number of transistors, if the block is manufactured with a smaller technology, let us say 45 nm, then its size will be divided by a $(180/45)^2$ factor, as shown in Fig. 1.4. Please note, that this factor is an approximation which is not always met with real physical design.

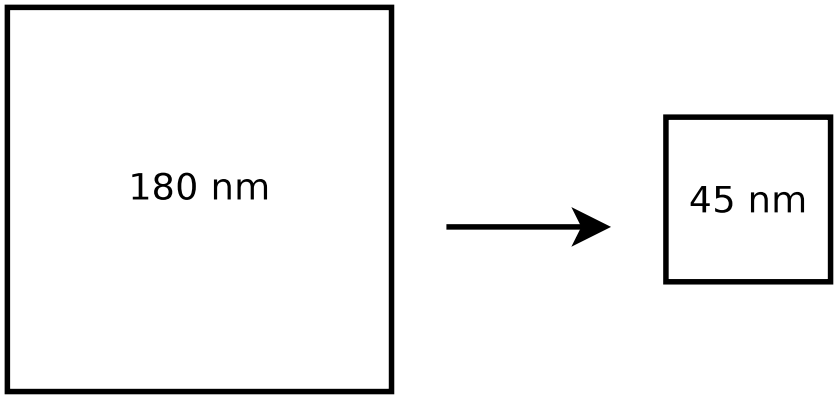

Figure 1.4: The use of different manufacturing technologies results in a size variation

Table 1.1: Scenario input matrix example

Component	ID	Size (90 nm)	Min aspect ratio	Size variability
ADRES 1~6	1~6	18.6 mm^2	0.5	0.1
FIFO	7	0.54 mm^2	0.5	0.1
L2D1-2	8-9	6.74 mm^2	0.5	0.1
L2Is1-1	10-11	6.62 mm^2	0.5	0.1
EMIF	12	0.66 mm^2	0.5	0.1
ARM	13	0.89 mm^2	0.5	0.1

ID: Component identification number

1.4 Case study and implementation

In this section, we will briefly explain the implementation of our design method and show some experimental results based on this first approach. We consider a case study based on the 3MF MPSoC platform developed at IMEC [6].

1.4.1 Description of the case study and modeling

The 3MF MPSoC is made of 13 blocks as shown on Fig. 1.5:
— 6 ADRES processors ([7])
— 2 data memories (L2D#)
— 2 instruction memories (L2Is#)
— 1 external memory interface (EMIF)
— 1 input/output processor (FIFO)
— 1 ARM processor (ARM)

Details about the area required for each component is given in Table 1.1 for a 90 nm technology. This table is also the input matrix required to specify the scenario.

The 3MF MPSoC can be configured for three use cases which have specific bandwidth requirements. For the following results, we will base our simulation on the "data split scenario" configuration which possesses the communication specifications shown in Table 1.2. This information is implemented, as shown in Table 1.3, in an input matrix which is built by specifying the communication structure: the first column will contain the ID of the source block and each next pairs of columns will contain the ID of the target blocks and the bandwidth required.

The input data are thus shown in Tables 1.1 and 1.3. The available technologies are also needed to take advantage of the heterogeneity. An example matrix for this input data is given in Table 1.4. Since no information is given about the ARM unit bandwidth usage, we will simplify our problem and not include it in the implementation. We consider therefore 12 blocks to assign.

Table 1.2: "Data split scenario" bandwidth requirements

Source	Target	Bandwidth (MB/s)
FIFO	EMIF	39.6
EMIF	ADRESi	6.6
L2D1	ADRESi	26.4
L2D2	L2D1	52.7
ADRESi	FIFO	1.2
ADRESi	L2D2	6.6
ADRESj	L2Is1	300
ADRESk	L2Is2	300

Index: $i, j, k \in \mathbb{N}^+$;
$1 \leq i \leq 6; 1 \leq j \leq 3; 4 \leq k \leq 6$

Table 1.3: Bandwidth input matrix

S	T	B	T	B	T	B	T	B	T	B	T	B
1	7	1.2	9	6.6	10	300	0	0	0	0	0	0
2	7	1.2	9	6.6	10	300	0	0	0	0	0	0
3	7	1.2	9	6.6	10	300	0	0	0	0	0	0
4	7	1.2	9	6.6	11	300	0	0	0	0	0	0
5	7	1.2	9	6.6	11	300	0	0	0	0	0	0
6	7	1.2	9	6.6	11	300	0	0	0	0	0	0
7	12	39.6	0	0	0	0	0	0	0	0	0	0
8	1	26.4	2	26.4	3	26.4	4	26.4	5	26.4	6	26.4
9	8	52.7	0	0	0	0	0	0	0	0	0	0
12	1	6.6	2	6.6	3	6.6	4	6.6	5	6.6	6	6.6

S: source block ID
(T, B): target block ID and required bandwidth

Table 1.4: Available technologies input matrix example

Technology (nm)				
90	60	45	32	22

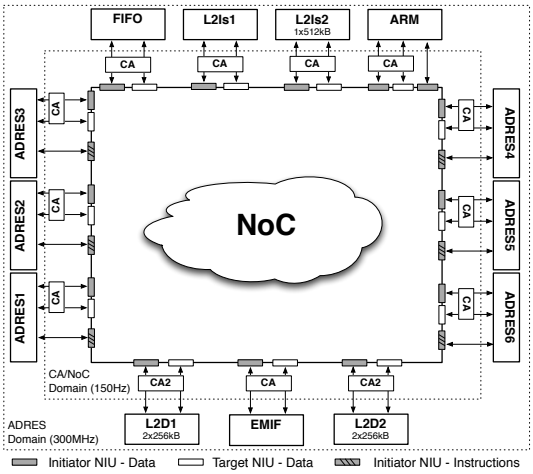

Figure 1.5: Architecture of the 3MF MPSoC platform [6]

In summary, the problem we consider is to place 12 blocks while taking into account several criteria (5). We will also consider a scenario where the blocks can be placed on 1 up to 5 tiers. The input data will be processed to generate floorplans. Those output data will be encoded using the matrix model following the example shown in Table 1.5. They will be generated through a multi-objective optimization.

As explained earlier, we will use a metaheuristic to approximate the Pareto optimal frontier. For that purpose, we choose to use NSGA-II [8]. The algorithm was run from a sample of 10 000 generated solutions from 1 up to 5 tiers. This size of random solutions is chosen arbitrarily since it is actually quite difficult to estimate the size of solution space, due to the heterogeneous nature of the criteria. As stated in Section 2, even for a really simplified problem, the solution space is huge (more than 10^{18}) so, taking 10 000, 100 000 or 1 000 000 randomly-generated solutions does not really imply any difference. Also, taking too few solutions (e.g. 100) is not interesting since we have empirically observed that our algorithm will take a longer time to begin to converge. 10 000 randomly-generated solutions seems to us a good compromise of time and workable solutions.

In the following section, we will present some details about the implementation of the NSGA-II algorithm.

1.4.2 Implementation of the exploration algorithm: NSGA-II

As shown in Table 1.5, we choose to encode our data in real or integer values, so that they can be used directly by design tools:

— The component identification number (ID) is a fixed integer value linked to

the component.
— The assigned layer (L) is a discrete value ranging from 1 to 5 in the case study.
— The geometrical coordinates (X,Y) are real values that depends on the dimension of the circuits and the aspect ratio of a block, so that the component cannot be placed outside the chip.
— The size (S) is a fixed real value linked to the component.
— The aspect ratio (AR) is a real value ranging from AR_{min} to $1/AR_{min}$ where AR_{min} is given as a specification as explained in Section 4.
— The length in X and Y axis (LX, LY) are real values computed from the size and the aspect ratio.
— The assigned technology per layer is a discrete value taking one of the specified technology (see Table 1.4).

This matrix will be our full chromosome for the NSGA-II algorithm.

We implemented our design space exploration following the steps of the NSGA-II which can be summarized by the diagram shown in Fig. 1.6.

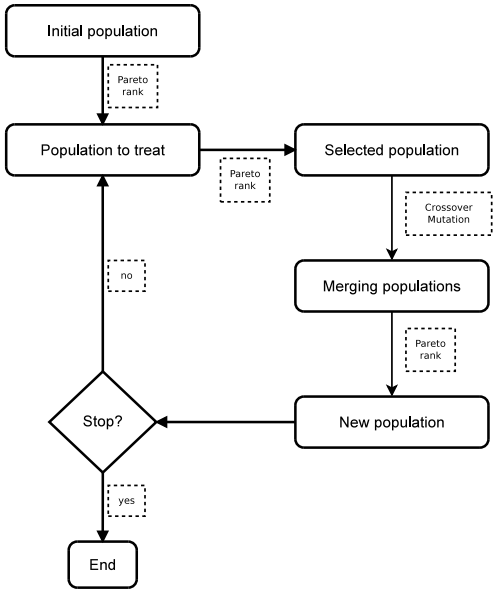

Figure 1.6: General NSGA-II steps

Initialization (the initial population)

We will work with a minimum size of population, namely 50, which is a common value in GAs [9]. The initial population will be a set of at least 50 solutions with the best Pareto ranks from a randomly-generated set of 10 000 solutions. The generated set places the blocks randomly (using a uniform distribution) and does not allow overlapping between the blocks. We could of course use a greedy algorithm as well as a more advanced method such as GRASP (Greedy Randomized Adaptive Search Procedure) [10]. This can be done as future work for comparison purposes.

Of course, having at least 50 Pareto solutions does not always happen. Actually, the selection is based on the Pareto rank so it does not include only the Pareto solutions (rank 1), but also the solutions with lower ranks until there are enough solutions.

Selection/crossover

For the selection step, two solutions will be allowed to make a crossover depending on a roulette wheel where the probability is proportional to the normalized Euclidean distance between the solutions ordered by their Pareto rank in the objective space. The normalization is done as follows :

$$\frac{g_i(a_j)}{\max\limits_{a_j \in A} g_i(a_j)} \tag{1.10}$$

where A is the set of alternatives with $a_j \in A$ and $g_i(a_j)$ is the evaluation of the alternative a_j on the criterion i.

The probability for two solutions to do a crossover will vary linearly with the Euclidean distance between them, as shown in Fig. 1.7. If two solutions are close to each other, they will have more chance to reproduce than if they are distant. This is to ensure the intensification properties of our algorithm. Therefore, we will have to specify a lower bound ($P_{c,min}$) and an upper bound ($P_{c,max}$) for the crossover probability. $P_{c,min}$ is set for the solutions which are the furthest to each other while $P_{c,max}$ is set for those which are the closest. In between, the probability will vary linearly inside these bounds.

These values will be fixed as $[P_{c,min} = 0.6; P_{c,max} = 1.0]$ since these seem to be common values [9].

Crossover

Let us now see how does the crossover occur. First, let us remark that it does not have limitations for the exploration process since the information contained in the matrix spans the whole circuit.

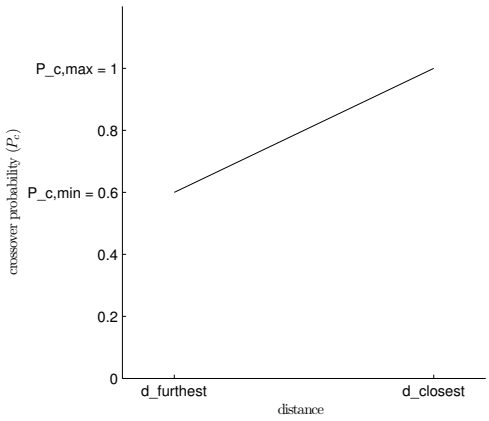

Figure 1.7: Evolution of the crossover probability as a function of the distance between two solutions

Second, we have to analyze how the chromosome is coded in order to see how we will apply the crossover step. For instance, let us choose the Layer (L) column as indicator for the crossover. If we order the matrix in Table 1.5 following this column, we will have the Table 1.6 and the Table 1.7 for another solution that we will use for the crossover.

Now, without loss of generality, let us suppose that the crossover happens (randomly) on line 7. One of the child will be the Table 1.8 and we see that the original scenario is not preserved since the first column (in bold) contains the same ID several times.

We observe that the only possible indicator for the crossover step is the ID column. Indeed if we order the two parents following the ID column, we have the Tables 1.9 and 1.10. If we still consider that the crossover occurs on line 7, we can have the child shown in Table 1.11. We see that there is no inconsistency since the scenario is still respected.

Mutation

A mutation cannot happen anywhere in the matrix. Indeed, if we take the conclusion about the choice of the crossover row indicator, all the elements except the ID column can mutate.

The mutation used is a random uniform distribution $U([a, b])$, where $[a, b]$ is the interval of values allowed for the mutation. For the discrete values, we use equidis-

Table 1.5: Output matrix template

ID	L	X	Y	S	AR	LX	LY	T
1	2	4.5	6	18.6	1	4.3128	4.3128	90
2	2	4	0.4	18.6	1	4.3128	4.3128	90
3	3	3.1	6.9	18.6	1	4.3128	4.3128	90
4	3	8.4	10.1	18.6	1	4.3128	4.3128	90
5	3	6.6	2.2	18.6	1	4.3128	4.3128	90
6	1	9	5.7	18.6	1	4.3128	4.3128	90
7	1	10	3.5	0.54	1	0.7348	0.7348	90
8	1	7.5	11	6.74	1	2.5962	2.5962	90
9	2	9	5	6.74	1	2.5962	2.5962	90
10	1	4.5	8	6.62	1	2.5729	2.5729	90
11	2	8.6	0.4	6.62	1	2.5729	2.5729	90
12	3	8.3	7.4	0.66	1	0.8124	0.8124	90

ID: component identification number; L: assigned layer;
(X, Y): geometrical coordinate; S: size (mm^2); AR: aspect ratio;
(LX, LY): length in X and Y axis; T: assigned technology for the layer

Table 1.6: First parent, ordered by L column; the line specifies the crossover cut

ID	L	X	Y	S	AR	LX	LY	T
6	1	9	5.7	18.6	1	4.3128	4.3128	90
7	1	10	3.5	0.54	1	0.7348	0.7348	90
8	1	7.5	11	6.74	1	2.5962	2.5962	90
10	1	4.5	8	6.62	1	2.5729	2.5729	90
1	2	4.5	6	18.6	1	4.3128	4.3128	90
2	2	4	0.4	18.6	1	4.3128	4.3128	90
9	2	9	5	6.74	1	2.5962	2.5962	90
11	2	8.6	0.4	6.62	1	2.5729	2.5729	90
3	3	3.1	6.9	18.6	1	4.3128	4.3128	90
4	3	8.4	10.1	18.6	1	4.3128	4.3128	90
5	3	6.6	2.2	18.6	1	4.3128	4.3128	90
12	3	8.3	7.4	0.66	1	0.8124	0.8124	90

ID: component identification number; L: assigned layer;
(X, Y): geometrical coordinate; S: size (mm^2); AR: aspect ratio;
(LX, LY): length in X and Y axis; T: assigned technology for the layer

Table 1.7: Second parent, ordered by L column; the line specifies the crossover cut

ID	L	X	Y	S	AR	LX	LY	T
4	1	1.6	8.5	18.6	1	4.3128	4.3128	90
5	1	1.2	1.3	18.6	1	4.3128	4.3128	90
6	1	0.6	4.7	18.6	1	4.3128	4.3128	90
3	2	5.9	4	18.6	1	4.3128	4.3128	90
9	2	5.4	8	6.74	1	2.5962	2.5962	90
10	2	8.5	8.1	6.62	1	2.5729	2.5729	90
11	2	2.8	4.6	6.62	1	2.5729	2.5729	90
1	3	7	6.3	18.6	1	4.3128	4.3128	90
2	3	7.4	9.8	18.6	1	4.3128	4.3128	90
7	3	5.6	5.5	0.54	1	0.7348	0.7348	90
8	3	2.8	5.5	6.74	1	2.5962	2.5962	90
12	3	5.7	7.5	0.66	1	0.8124	0.8124	90

ID: component identification number; L: assigned layer;
(X, Y): geometrical coordinate; S: size (mm^2); AR: aspect ratio;
(LX, LY): length in X and Y axis; T: assigned technology for the layer

Table 1.8: Possible child, ordered by L column

ID	L	X	Y	S	AR	LX	LY	T
6	1	9	5.7	18.6	1	4.3128	4.3128	90
7	1	10	3.5	0.54	1	0.7348	0.7348	90
8	1	7.5	11	6.74	1	2.5962	2.5962	90
10	1	4.5	8	6.62	1	2.5729	2.5729	90
1	2	4.5	6	18.6	1	4.3128	4.3128	90
2	2	4	0.4	18.6	1	4.3128	4.3128	90
9	2	9	5	6.74	1	2.5962	2.5962	90
1	3	7	6.3	18.6	1	4.3128	4.3128	90
2	3	7.4	9.8	18.6	1	4.3128	4.3128	90
7	3	5.6	5.5	0.54	1	0.7348	0.7348	90
8	3	2.8	5.5	6.74	1	2.5962	2.5962	90
12	3	5.7	7.5	0.66	1	0.8124	0.8124	90

ID: component identification number; L: assigned layer;
(X, Y): geometrical coordinate; S: size (mm^2); AR: aspect ratio;
(LX, LY): length in X and Y axis; T: assigned technology for the layer

Table 1.9: First parent, ordered by ID column; the line specifies the crossover cut

ID	L	X	Y	S	AR	LX	LY	T
1	2	4.5	6	18.6	1	4.3128	4.3128	90
2	2	4	0.4	18.6	1	4.3128	4.3128	90
3	3	3.1	6.9	18.6	1	4.3128	4.3128	90
4	3	8.4	10.1	18.6	1	4.3128	4.3128	90
5	3	6.6	2.2	18.6	1	4.3128	4.3128	90
6	1	9	5.7	18.6	1	4.3128	4.3128	90
7	1	10	3.5	0.54	1	0.7348	0.7348	90
8	1	7.5	11	6.74	1	2.5962	2.5962	90
9	2	9	5	6.74	1	2.5962	2.5962	90
10	1	4.5	8	6.62	1	2.5729	2.5729	90
11	2	8.6	0.4	6.62	1	2.5729	2.5729	90
12	3	8.3	7.4	0.66	1	0.8124	0.8124	90

ID: component identification number; L: assigned layer;
(X, Y): geometrical coordinate; S: size (mm^2); AR: aspect ratio;
(LX, LY): length in X and Y axis; T: assigned technology for the layer

Table 1.10: Second parent, ordered by ID column; the line specifies the crossover cut

ID	L	X	Y	S	AR	LX	LY	T
1	3	7	6.3	18.6	1	4.3128	4.3128	90
2	3	7.4	9.8	18.6	1	4.3128	4.3128	90
3	2	5.9	4	18.6	1	4.3128	4.3128	90
4	1	1.6	8.5	18.6	1	4.3128	4.3128	90
5	1	1.2	1.3	18.6	1	4.3128	4.3128	90
6	1	0.6	4.7	18.6	1	4.3128	4.3128	90
7	3	5.6	5.5	0.54	1	0.7348	0.73485	90
8	3	2.8	5.5	6.74	1	2.5962	2.5962	90
9	2	5.4	8	6.74	1	2.5962	2.5962	90
10	2	8.5	8.1	6.62	1	2.5729	2.5729	90
11	2	2.8	4.6	6.62	1	2.5729	2.5729	90
12	3	5.7	7.5	0.66	1	0.8124	0.8124	90

ID: component identification number; L: assigned layer;
(X, Y): geometrical coordinate; S: size (mm^2); AR: aspect ratio;
(LX, LY): length in X and Y axis; T: assigned technology for the layer

Table 1.11: Possible child, ordered by ID column

ID	L	X	Y	S	AR	LX	LY	T
1	2	4.5	6	18.6	1	4.3128	4.3128	90
2	2	4	0.4	18.6	1	4.3128	4.3128	90
3	3	3.1	6.9	18.6	1	4.3128	4.3128	90
4	3	8.4	10.1	18.6	1	4.3128	4.3128	90
5	3	6.6	2.2	18.6	1	4.3128	4.3128	90
6	1	9	5.7	18.6	1	4.3128	4.3128	90
7	1	10	3.5	0.54	1	0.7348	0.7348	90
8	3	2.8	5.5	6.74	1	2.5962	2.5962	90
9	2	5.4	8	6.74	1	2.5962	2.5962	90
10	2	8.5	8.1	6.62	1	2.5729	2.5729	90
11	2	2.8	4.6	6.62	1	2.5729	2.5729	90
12	3	5.7	7.5	0.66	1	0.8124	0.8124	90

ID: component identification number; L: assigned layer;
(X, Y): geometrical coordinate; S: size (mm^2); AR: aspect ratio;
(LX, LY): length in X and Y axis; T: assigned technology for the layer

tributed probabilities. The mutation probability of a child will be set as $P_m = 0.3$. Empirical observations have shown that smaller mutation probability can easily lead to a local optimum. This can be explained by the fact that we choose that only one single element of a line can mutate instead of the whole line. If a child is forced to mutate, then one randomly-chosen value of the whole matrix will mutate within the range of values it is allowed to take.

A Gaussian mutation is also a common operator but it has not been chosen since it will produce a solution which is not far from the original one. This is not really interesting to have similar solutions when exploring the design space for integrated circuits. Of course, a large standard deviation value can be chosen but this will be likely to produce solution which are out of the feasible bounds.

Consistency test

Of course, infeasible solutions may appear after the crossover/mutation step, since these operations are made with randomness. In order to verify that, we perform a test on each new solution to check if there is overlapping between the blocks. Currently, the solutions which are infeasible will be discarded. Of course, it is possible to apply some repair mechanism but it is to be investigated as future work even if we already produce feasible solutions.

Termination

Three stop conditions have been implemented and are based on what is commonly used:

— Maximum number of iterations, set to 100.
— Maximum elapsed time, set to 30 minutes.
— Maximum number of iterations with an unchanged population, set to 10.

The maximum elapsed time has been chosen arbitrarily for quick testing purposes. Having a simulation time of a few hours would not be a problem either. Indeed, in practice, the optimization of one single architecture can take from several hours to several days with the current design tools.

1.5 Results and their use for a designer

The optimization was done for three criteria (so that we can visualize the design space) and the main results are given in Fig. 1.8 (3D plot) and Fig. 1.9 (interconnection length-cost projection). Two conclusions can be drawn from that figure:

— The [10; 20] range values for the IL criteria: a small enhancement of the IL value leads to a large increase of the cost so the interest for a design with more than 4 tiers seems low.
— The [260; 280] range values for the cost criteria: a small increase of the price can give a large enhancement of the performance. A designer might consider accepting a slightly higher price for a sensitively better performance, knowing that this information can be quantified with an accurate model. Indeed, with the estimate model that we propose, a small 10% increase of the cost can decrease the IL by 60%.

These results from did not take into account the degree of freedom of aspect ratio. If we go further by releasing a degree of freedom and allowing varying aspect ratios, we can have the Pareto front shown in Fig. 1.10. This figure shows the Pareto front from Fig. 1.9 (without aspect ratio, symbol: $\cdot$) alongside with a new Pareto front (with aspect ratio, symbol: +).

As expected, the Pareto front given when considering varying aspect ratios is globally better. Furthermore, by comparing the two graphs, we can see an interesting area where the two frontiers begin to merge at the cost value 350. This means that, in that area, it is not necessary to take the aspect ratio into account. Once again, these kind of information can be important in the design of an IC and yet they would not be available with the current design flows since only a small number of possibilities are explored. Indeed, due to the sequential nature of the current design flows, such degrees of freedom are not even tried since they dramatically increase the duration of each optimization loop.

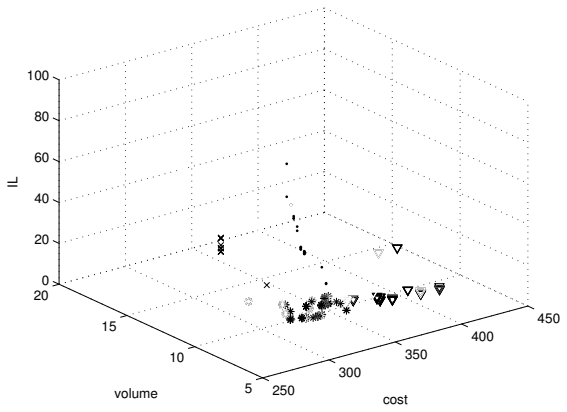

Figure 1.8: 3D view (interconnection length-volume-cost) of the Pareto frontier

· : 1 tier; × : 2 tiers; + : 3 tiers; * : 4 tiers; ▽ : 5 tiers

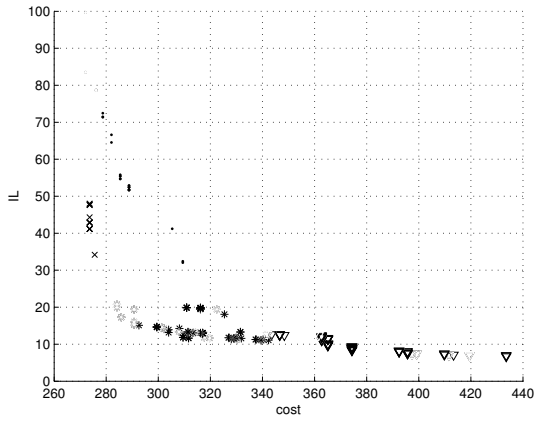

Figure 1.9: IL-cost projection view of the Pareto frontier

· : 1 tier; × : 2 tiers; + : 3 tiers; * : 4 tiers; ▽ : 5 tiers

1.6 Conclusion

The results have thus shown interesting analyses that can be relevant for a designer. First, using a multi-objective optimization methodology does not only consider all the criteria at the same time but also proceed to an extensive design space

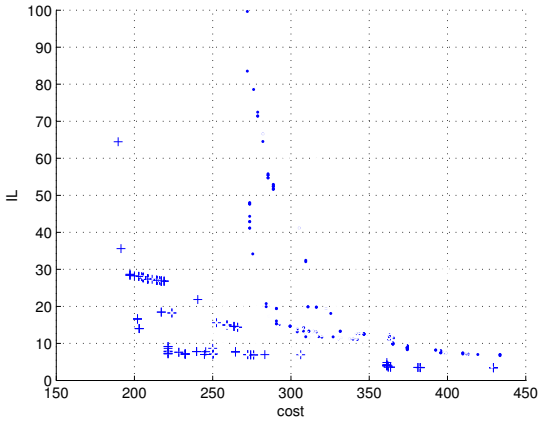

Figure 1.10: IL-cost projection view of the Pareto frontier (with and without aspect ratio)

· : Pareto front without aspect ratio; + : Pareto front with aspect ratio

exploration which is rarely done with current tools. Second, the qualitative results shown here can give relevant information to the designer and they can be quantified with a more accurate model. Third, the flexibility of MOO allows to easily consider new degrees of freedom without having to change the paradigm. Finally, this methodology and the associated algorithms has shown positive indicators of convergence and robustness as it will be shown in the next chapter.

2

Robustness of the methodology

In chapter 1, we have shown how a 3D-SIC can be modelled in order to apply a NSGA-II metaheuristic. The obtained results indicate that multi-objective optimization can give qualitative and quantitative information to a designer that would not be available with current design tools.

In this chapter we will take a deeper look at the results of the multi-objective optimization. We will analyze the properties of the design space in order to have the view over the convergence and the robustness of our methodology. We will use classical performance indicators such as the contribution indicator, the spread indicator, the binary ϵ-indicator, the unary hypervolume indicator and the density of the Pareto-front which are presented in [11, 12]. These indicators can be grouped in 3 categories defined in [11]:

1. The convergence-based indicators:

   "The convergence metrics evaluate the effectiveness of the solutions in terms of the closeness to the optimal Pareto front."

2. The diversity-based indicators:

   "Diversity indicators measure the uniformity of distribution of the obtained solutions in terms of dispersion and extension. In general, the diversity is researched in the objective space."

3. The hybrid indicators: that combine both convergence and diversity measures.

2.1 Contribution indicator

The contribution is a convergence-based binary indicator. The contribution of an approximation PO_1 relatively to another approximation PO_2 is the ratio of non-dominated solutions produced by PO_1 in PO^*, which is the set of Pareto solutions of $PO_1 \cup PO_2$:

$$Cont(PO_1/PO_2) = \frac{\frac{\|PO\|}{2} + \|W_1\| + \|N_1\|}{\|PO^*\|} \tag{2.1}$$

where PO is the set of solutions in $PO_1 \cap PO_2$, W_1 the set of solutions in PO_1 that dominate some solutions of PO_2 and N_1 the set of non-comparable solutions of PO_1. This value has to be greater than 0.5 to indicate that PO_1 is better than PO_2 in terms of convergence to the Pareto front.

The Table 2.1 and the Fig. 2.1 show the evolution of the averaged contribution indicator over the iterations for the 5 experiments. We see that for the first iterations, $Cont(PO_i/PO_{i-1})$ is greater than 0.5, which means that the algorithm does indeed improve the solutions, then for the last iterations, the indicators are lower than 0.5 which means that there is a convergence.

Iteration	$Cont(PO_i/PO_{i-1})$
1	0.7626
2	0.8510
3	0.8917
4	0.8788
5	0.8295
...	...
38	0.4522
39	0.3870
40	0.2369

Table 2.1: Evolution of the contribution indicator

2.2 Spread indicator

The spread indicator I_s combines the distribution and cardinality to measure the dispersion of the approximated set A:

$$I_s = \frac{\sum_{u \in A} |\{u' \in A : \|F(u) - F(u')\| > \sigma\}|}{|A| - 1} \tag{2.2}$$

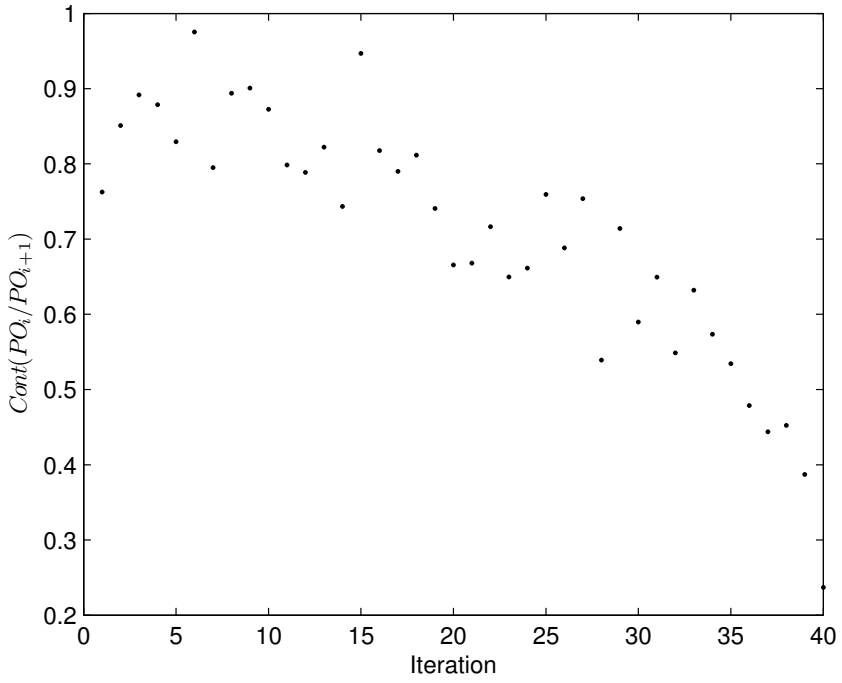

Figure 2.1: Evolution of the contribution indicator

where $F(u)$ is a fitness function and $\sigma > 0$ a neighborhood parameter. The closer is the measure to 1, the better is the spread of the approximated set A.

The Fig. 2.2 shows the results of the spread indicator I_s function of the neighborhood indicator σ (all the 5 experiments share the same graph shape). We see that the Pareto front is well spread ($I_s \geq 0.9$) for $\sigma < 0.35$ in average for the 5 runs (normalized values), so we can consider that the algorithm produces a well-spread approximation of the Pareto front.

2.3 Binary ϵ-indicator

The binary ϵ-indicator is a convergence-based indicator. It will give the quality of a solution front in comparison with a another set, with regards to all objectives. Let us consider a minimization problem with n positive objectives. An objective vector $f^1 = (z_1^1, z_2^1, \ldots, z_n^1)$ is said to ϵ-dominate another objective vector $f^2 = (z_1^2, z_2^2, \ldots, z_n^2)$ if $\forall 1 \leq i \leq n : \ z_i^1 \leq \epsilon \cdot z_i^2$, for a given $\epsilon > 0$. A binary ϵ-indicator $I_\epsilon(A, B)$ gives the factor ϵ such that for any solution in B there is at least one solution in A that is not worse by a factor of ϵ in all objectives. $I_\epsilon(A, B)$ can be calculated as

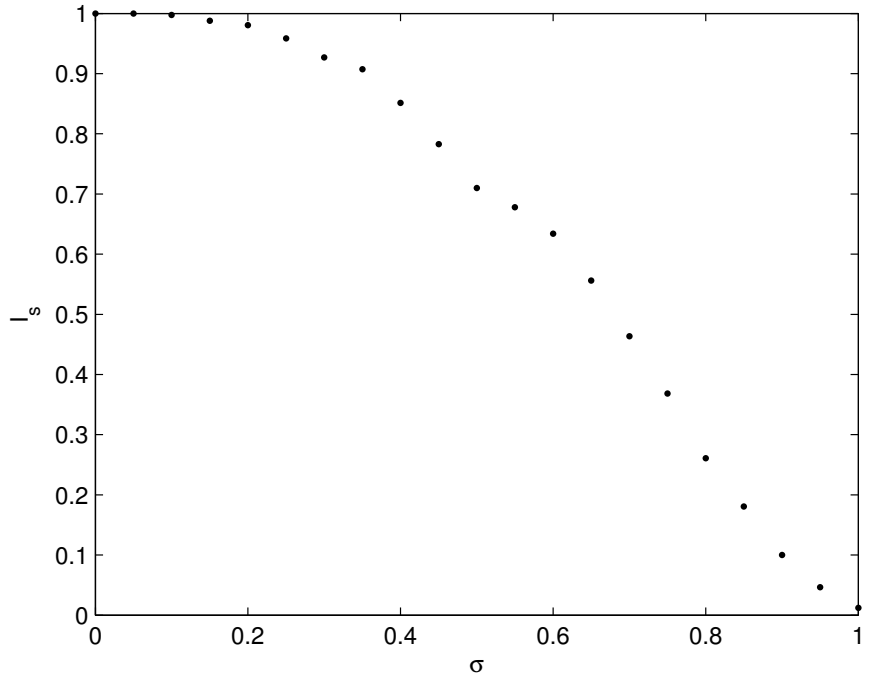

Figure 2.2: Spread indicator I_s function of the neighborhood parameter σ

follows [12]:

$$I_\epsilon(A,B) = \max_{z^2 \in B} \min_{z^1 \in A} \max_{1 \leq i \leq n} \frac{z_i^1}{z_i^2} \tag{2.3}$$

The reference set R computed from all the experiments will serve to show the evolution of the ϵ-indicator over time. This evolution is shown in Table 2.2 (for averaged values after each 10 iterations). We can see that in the first iterations, $I_\epsilon(A,R) > 1$ and $I_\epsilon(R,A) \approx 1$ which means that the front is improved while in the last iterations, $I_\epsilon(A,R) > 1$ and $I_\epsilon(R,A) > 1$ which shows convergence.

A comparison of the binary ϵ-indicators between each experiment is also given, in Table 2.3. We can see that $I_\epsilon(A,B) > 1$ and $I_\epsilon(B,A) > 1$ which indicates that neither A weakly dominates B nor B weakly dominates S. This means that the generated front is consistent from one experiment to another.

Also, in Table 2.4 are given the ϵ-indicator between iterations of an experiment (the same observations apply for the other runs). We can see that in the first iterations, the front is always improved ($I_\epsilon(A_i, A_{i-1}) > 1$ and $I_\epsilon(A_{i-1}, A_i) \leq 1$) while in the last iterations, it begins to converge ($I_\epsilon(A_i, A_{i-1}) > 1$ and $I_\epsilon(A_{i-1}, A_i) > 1$).

2.4 Unary hypervolume indicator

The hypervolume is an hybrid indicator. Since we already used a binary indicator (*epsilon*), we will use the hypervolume indicator I_H in its unary form. I_H, associated with an approximation set A is given by the volume of the space portion that is weakly dominated by the set A [11].

The evolution of the hypervolume (averaged values) is given in Table 2.5 and in Figure 2.3. We can see that the value is (linearly) increasing over time.

The result for each experiment is also given, in Table 2.6 and the used reference point is the worst point computed from all the sets. As we can see, the values are rather consistent from one run to another.

2.5 Density of the Pareto front - gaps in the frontier

Another indicator of the Pareto front structure is its density. Here we will measure the density by finding gaps in the frontier. This will be done by counting the number of solutions in the neighborhood of another solution. Since the extreme distance between two solutions is 450.364 in average for the 5 runs (non-normalized), we consider that an acceptable neighborhood is twice the distance between two solutions if all the solutions were equidistant. We have thus a neighborhood of about 2. This test has shown that there was always at least one solution near another one, even for a neighborhood of 1, meaning that the algorithm can produce a sufficiently dense frontier.

Iteration	Averaged $I_\epsilon(A, R)$	Averaged $I_\epsilon(R, A)$
1	5.5255	1.0178
10	4.4307	1.0235
20	3.8102	1.1023
30	2.3614	1.1234
40	1.6569	1.2381

Table 2.2: Evolution of the binary ϵ-indicator (averaged values compared to the reference set R) over time

2.6 Conclusion

In this chapter, we have shown that the used methodology has proved to be robust even if the problem contains criteria of heterogeneous nature. With the several indicators that we have analysed, we can conclude that the algorithm we used can show good properties of convergence, spread and density.

$I_\epsilon(A, B)$	Run 1	Run 2	Run 3	Run 4	Run 5
Run 1	1	1.7270	1.3594	1.8664	1.2542
Run 2	1.5713	1	1.4122	1.7791	1.3420
Run 3	1.4737	1.8638	1	1.9268	1.3069
Run 4	1.3436	1.4564	1.2843	1	1.2365
Run 5	1.4214	1.7650	1.3918	1.7545	1

Table 2.3: Comparison of the binary ϵ-indicators for each experiment

Iteration	$I_\epsilon(A_i, A_{i-1})$	$I_\epsilon(A_{i-1}, A_i)$
1	1.6674	1.1053
2	1.7223	1
3	2.4439	1
4	1.7477	1
5	2.0577	1
...	...	...
38	1.8788	1.4916
39	1.5344	1.8065
40	1.9862	1.6609

Table 2.4: Comparison of the binary ϵ-indicators between iterations of the same experiment

Iteration	Averaged hypervolume
1	0.0574
10	0.0701
20	0.0876
30	0.0931
40	0.1036

Table 2.5: Evolution of the unary hypervolume indicator (averaged values compared to the reference set R) over time

The methodology can thus be considered as robust even if the problem is not homogeneous (heterogeneous nature of the criteria) and this shows that a multi-criteria paradigm can be suitable for the design of 3D-SICs.

Also, analyses have been performed to determine the shape of the Pareto front. Globally, the Pareto front is not convex, as one may expect since the heterogeneous nature of the criteria. This is probably due to some correlation between the criteria but this is still to be investigated.

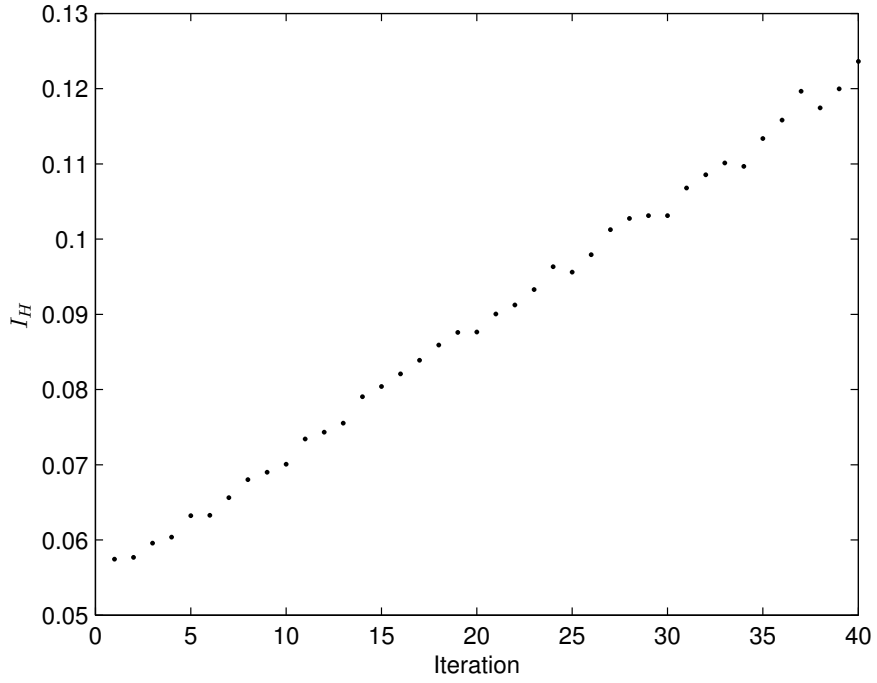

Figure 2.3: Evolution of the unary hypervolume indicator (averaged values compared to the reference set R over time

Run	I_H
1	0.1171
2	0.1105
3	0.1015
4	0.1185
5	0.0939

Table 2.6: Hypervolume for each experiment

Bibliography

[1] D. Milojevic, T. Carlson, K. Croes, R. Radojcic, D. F. Ragett, D. Seynhaeve, F. Angiolini, G. V. der Plas, and P. Marchal, "Automated pathfinding tool chain for 3D-stacked integrated circuits: Practical case study." in *3DIC*. IEEE, 2009, pp. 1–6.

[2] J. Cong, J. Wei, and Y. Zhang, "A thermal-driven floorplanning algorithm for 3D ICs," in *ICCAD '04: Proceedings of the 2004 IEEE/ACM International conference on Computer-aided design*. Washington, DC, USA: IEEE Computer Society, 2004, pp. 306–313.

[3] H. Yan, Q. Zhou, and X. Hong, "Thermal aware placement in 3D ICs using quadratic uniformity modeling approach," *Integr. VLSI J.*, vol. 42, no. 2, pp. 175–180, 2009.

[4] G. Pelosi, "The finite-element method, part i: R. l. courant [historical corner]," *Antennas and Propagation Magazine, IEEE*, vol. 49, no. 2, pp. 180 –182, april 2007.

[5] N. Doan, F. Robert, Y. D. Smet, and D. Milojevic, "Mcda-based methodology for efficient 3d-design space exploration and decision," *International Symposium on System-on-Chip Proceedings (SoC 2010)*, pp. 76–83, September 2010.

[6] D. Milojevic, L. Montperrus, and D. Verkest, "Power dissipation of the network-on-chip in multi-processor system-on-chip dedicated for video coding applications," *Journal of Signal Processing Systems*, p. 15, June 2008.

[7] F.-J. Veredas, M. Scheppler, W. Moffat, and B. Mei, "Custom implementation of the coarse-grained reconfigurable adres architecture for multimedia purposes." in *FPL*, T. Rissa, S. J. E. Wilton, and P. H. W. Leong, Eds. IEEE, 2005, pp. 106–111.

[8] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multi-objective genetic algorithm: NSGA-II," *IEEE Transactions on Evolutionary Computation*, vol. 6, pp. 182–197, 2000.

[9] L. Davis, "Adapting operator probabilities in genetic algorithms," in *Proceedings of the third international conference on Genetic algorithms*. San

Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1989, pp. 61–69. [Online]. Available: http://dl.acm.org/citation.cfm?id=93126.93146

[10] J. Hart and A. Shogan, "Semi-greedy heuristics: An empirical study," *Operations Research Letters*, vol. 6, pp. 107–114, 1987.

[11] E.-G. Talbi, *Metaheuristics : from design to implementation*. John Wiley & Sons, 2009.

[12] E. Zitzler, L. Thiele, M. Laumanns, C. Fonseca, and V. da Fonseca, "Performance assessment of multiobjective optimizers: an analysis and review," *Evolutionary Computation, IEEE Transactions on*, vol. 7, no. 2, pp. 117 – 132, april 2003.