1

MCDM applied to the partioning problem of 3D-stacked integrated circuits

1.1 Introduction

In order to continuously improve the performance of integrated circuits (IC), technologists have compelled themselves to follow the well-known Moore's Law (see Figure 1.1). This empirical law predicts a doubling of the transistors' density each 18 months and therefore increasing logic capacity of the circuit per unit area.

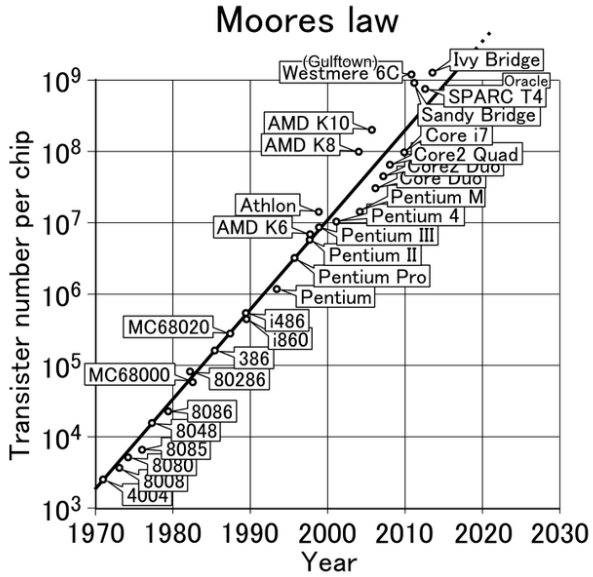

Fig. 1.1. Moore's law [4]

The improvements of 2D architectures are primarily driven by the reduction of the transistor size. However, with the miniaturization, quantum effects such as quantum tunnelling will significantly affect how a transistor behave. Indeed, even if a transistor is blocking, current can flow through due to quantum tunnelling such that it will be difficult to control its state and thus the basic working principle of a transistor [14]. In addition to these physical aspects, economical considerations that will hinder the IC evolution beyond 20nm have to be taken into account [3, 11].

In order to overcome these limitations, new technologies have been proposed such as the carbon nanotubes [13], the nanowire transistors [5], the single-electron transistors [8], and also the 3D-Stacked Integrated Circuits (3D-SIC) proposed by the academic and industrial communities. The latter has been often cited as the most prominent one as it is based on the current technologies and still uses silicon as basis material; 3D-SICs can also allow shorter interconnection lengths, smaller footprint, larger bandwidth, heterogeneous circuits among their main advantages [1, 12, 6, 2].

Fast evolution of IC manufacturing technologies makes even the design of 2D-ICs a complex and tedious task with the growing number of design choices at the system level (e.g. number and type of functional units and memories, type and topology of the interconnection system, etc.) and physical level (respecting area/timing/power constraints). Using 3D-SICs introduces even more degrees of freedom: number of tiers, choices for manufacturing technology (e.g. full 3D integration, silicon interposer, face-to-face, back-to-face, etc.), 3D partitioning and placement strategies etc. These new degrees of freedom will contribute to the combinatorial explosion of already huge design spaces. Moreover, practice and 2D design experience cannot be fully exploited with 3D technology, since 3D-SICs change considerably the way ICs are implemented. Indeed, physical implementation of ICs involves solving several complex problems and hence work only with approximated solutions.

Current design flows can produce workable solutions after manual definition of the physical constraints as there are no preconceived method that can provide good solutions. Also, they are sequential in nature as certain parameters are fixed at certain stages in the flow, which can lead to locally optimal solutions that are far from global optimums so this requires time consuming (hence, costly) iterative processes to adjust these parameters. Since the 3D technology is even more complex than the 2D, it is necessary to improve the current design flows by developing design exploration [11].

One of the solutions to face this problem is to develop high-level tools which can quickly explore design spaces and give early and reasonably accurate performance estimations based on physical prototyping of the 3D circuits [11]. In addition, performance estimation/optimization and the selection of the most-suitable solutions usually implies to take several objectives into account (e.g. maximization of the performance, minimization of the cost, minimization of the package size, etc.).

Currently, these high-level design tools can be considered to follow a unicriterion paradigm. Indeed, they have sequential development steps and each criterion is optimized without considering the impact on other criteria. This can lead to several rollbacks in the design flow since the achievement of the requirements can be time consuming (typical design iterations are measured in weeks). For instance, current tools will only minimize the area of a circuit to reach the timing constraints by solving a 2D place-and-route problem and this will be more complex with 3D-SICs because the system has also to be partitioned.

On the other hand, multi-objective approaches have been developed to optimize all the criteria simultaneously. Designing 3D-SICs inherently implies a huge design space and numerous degrees of freedom and criteria, hence many possible choices when it comes to decide upon the IC to produce. With these reasons, we propose in this work to apply a multi-criteria paradigm with the PROMETHEE methods for the design of 3D-SICs.

1.2 Related works

1.2.1 Multi-criteria decision making tools: using the PROMETHEE methods

In this subsection we recall the basics of the PROMETHEE and GAIA methods. Of course, a detailed description of these approaches goes beyond the scope of this contribution. Therefore we refer the interested reader to [7] for a detailed analysis.

Let $\mathcal{A} = \{a_1, a_2, \ldots, a_n\}$ be a set of n alternatives and $\mathcal{F} = \{f_1, f_2, \ldots, f_m\}$ be a set of m criteria. Without loss of generality, we assume that all criteria have to be maximized. The PROMETHEE methods are based on pairwize comparisons. At first, each pair of alternatives $a_i, a_j \in \mathcal{A}$ is compared on every criterion f_k:

$$d_k(a_i, a_j) = f_k(a_i) - f_k(a_j)$$

The quantity $d_k(a_i, a_j)$ represents the *advantage* of a_i over a_j for criterion f_k. On the one hand, when $d_k(a_i, a_j)$ is small enough, there is no good reason to say that a_i is better than a_j regarding criterion f_k. On the other hand, when $d_k(a_i, a_j)$ exceeds a certain limit, the decision maker may express that a_i is strictly preferred to a_j for f_k. In order to model these statements, the difference $d_k(a_i, a_j)$ is transformed into a unicriterion preference degree, denoted $P_k(a_i, a_j)$, by using a non-decreasing function H_k;

$$P_k(a_i, a_j) = H_k(d_k(a_i, a_j)), \ \ \forall\, a_i, a_j \ \in \ \mathcal{A}$$

The quantity $P_k(a_i, a_j) \in [0, 1]$ and $P_k(a_i, a_j) = 0$ when $d_k(a_i, a_j) < 0$. There are plenty of functions that can be considered to compute the unicriterion preference degrees. In most software implementing the PROMETHEE

method, 6 main functions are considered [9]. Figure 1.2 represents the so-called linear preference function. Two thresholds characterize it:

- q_k plays the role of an *indifference* threshold. When the difference $d_k(a_i, a_j) \leq q_k$, it is considered to be so small that the unicriterion preference is equal to zero;
- p_k plays the role of a *preference* threshold, When the difference $d_k(a_i, a_j) \geq p_k$, it is considered to be important enough to state that a_i is strongly preferred to a_j for this criterion.

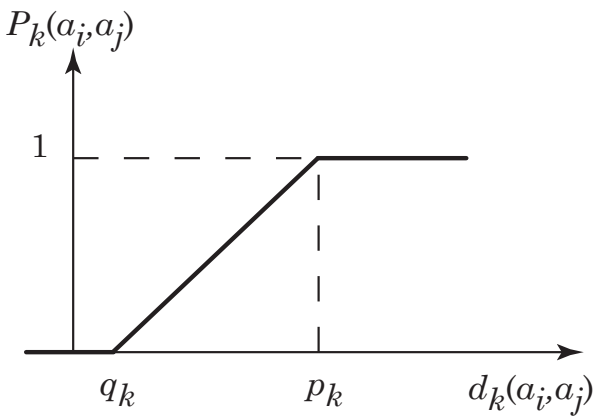

Fig. 1.2. Generalized criterion of type 5

Once the unicriterion preference degrees between two actions a_i and a_j have been computed for every criterion, one has to aggregate these marginal contributions to obtain $P(a_i, a_j)$ i.e. a global measure of the preference of a_i over a_j:

$$P(a_i, a_j) = \sum_{k=1}^{m} \omega_k \cdot P_k(a_i, a_j)$$

where ω_k represents the relative importance of criterion f_k. These weights are assumed to be positive and normalized. Obviously, we have $P(a_i, a_j) \geq 0$ and $P(a_i, a_j) + P(a_j, a_i) \leq 1$.

The PROMETHEE I and II rankings are based on the exploitation of the matrix P. Therefore, three flows are built.; the positive flow ϕ^+, the negative flow ϕ^- and the net flow ϕ:

$$\phi^+(a_i) = \frac{1}{n-1} \sum_{a_j \in \mathcal{A}, i \neq j} P(a_i, a_j)$$

$$\phi^-(a_i) = \frac{1}{n-1} \sum_{a_j \in \mathcal{A}, i \neq j} P(a_j, a_i)$$

$$\phi(a_i) = \phi^+(a_i) - \phi^-(a_j)$$

The PROMETHEE I ranking is obtained as the intersection of the rankings induced by ϕ^+ and ϕ^-. The PROMETHEE II ranking is given by the ranking given by ϕ.

Finally, it is worth noting that:

$$\phi(a_i) = \frac{1}{n-1} \sum_{k=1}^{m} \sum_{a_j \in \mathcal{A}} [P_k(a_i, a_j) - P_k(a_j, a_i)] \cdot \omega_k = \sum_{k=1}^{m} \phi_k(a_i) \cdot \omega_k$$

where $\phi_k(a_i)$ is called the k^{th} unicriterion net flow assigned to action a_i.

The PROMETHEE I and II ranking provide prescriptive tools for decision making. The GAIA [10] tool complements them with a descriptive approach. The idea is to represent each alternative by its evaluations in the unicriterion net flow space:

$$\Phi(a_i) = [\phi_1(a_i), \phi_2(a_i), \ldots, \phi_m(a_i)]$$

GAIA is the result of a principal component analysis applied on this dataset. Therefore, the decision maker is able to visualize the decision problem on a plane and compare:

- the relative positions of alternatives (in order to identify groups of similar or distinct alternatives profiles);
- the relative positions of criteria (in order to identify conflicts or redundancies);
- the relative positions of alternatives with respect to a given criterion (in order to identify the best and worst alternatives for the different points of views);
- the relative positions of alternatives with respect to the so-called *decision stick* (in order the identify the best compromise solutions).

1.2.2 3D integration technology

3D integration technology is considered to be one of the most promising paths to enable further scaling of Integrated Circuits (ICs). Over the past years many different 3D integration technology flavors have been proposed in both academia and industry. Depending on the integration granularity, a very coarse grain 3D technology classification differentiates between 3D-Stacked Integrated Circuits (3D-SICs) and 3D Monolithic integration. Each of these technology options has it's own merits and drawbacks and as of today there is

no clear preference for one or the other. The right choice is very much design dependent and is strongly affected by our ability to perform optimal design implementation.

3D-SICs are built using 2 (or more) fully processed integrated circuits (wafers) that are integrated vertically one on the top of each other (i.e. dies are stacked). Different 3D structures that enable this particular type of 3D integration have been proposed over the past years and they typically include: wire bonding, Through Silicon Vias (TSV), micro-bumps and copper pads. Even if the wire bonding technique is well known, practical usage remained limited because of the connection pitch that is quite high ($\tilde{1}$00 micrometer range). Also, wire bonds have high resistance and capacitance that will strongly affect delay and power of a 3D wire. Further, as wire bonds appear at the periphery of the IC, the signals connected to them need to be routed throughout the whole circuit. For all these reasons it is generally considered that the wire bonding is not well suited for efficient 3D integration.

On the other hand TSVs, micro-bumps and copper pads are much more promising technology features since they can be manufactured at very low pitch, the diameter depends on the structure but is generally in the um range allowing dense 3D integration (many inter-die nets). These 3D structures also have good resistance and capacitance values, allowing small delay and power overheads of 3D nets compared to wire bonding and even 2D as long as we enable some wirelength savings for 3D.

With these structures 3D circuits can be stacked in many different ways depending on the orientation of the circuit *face*, i.e. the side of the IC where we find the active layer (transistors i.e. gates). Not all of the options are interesting from the integration perspective, Face-to-Face (F2F) and Face-to-Back (F2B) integration schemes are the ones that are used most of the time. Cross-sections of F2F and F2B 3D integration schemes is shown on Figure 1.3).

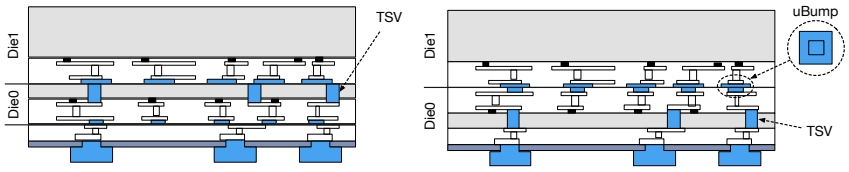

Fig. 1.3. Cross section of Face-to-Back and Face to Face 3D-Stacked Integrated Circuit

In F2F 3D-SIC, the face of both dies are oriented towards each other and they are interconnected directly. Input/output TSVs are used to connect the active layer of one of the dies to the package ensuring the system communication with the external world. F2F is in principal limited to stacking of two dies only (although it would be possible to stack yet another die on the

top of the stack using F2B approach). In case of F2B 3D-SIC, face of one die is oriented towards the back of other die. The active layers of the dies are connected using TSVs, a vertical connection that goes through the substrate of the die. Active layer of one of the dies is exposed to the package used for communication of the system with the external world. F2B can be used to stack more than two dies and is used for manufacturing of highly dense SRAM and DRAM circuits.

The assembly of the dies can be carried out at die or at the wafer level, hence we distinguish: a) Die-to-Die, b) Die-to-Wafer, or c) Wafer-to-Wafer 3D integration. Each assembly method has its advantages (and disadvantages) and the choice depends on the application needs to realize the cost benefits of the 3D integration (trade-off between the wafer processing speed, yield and die area). From the design perspective, there is no significant difference in the choice of assembly approach. We can thus safely assume any of the proposed assembly methods. 3D stacking technology is mature today and is already used in production lines (e.g. DRAM/FLASH memories).

In 3D monolithic integration multiple transistor or logic gate layers are formed sequentially starting from the bottom-most layer. Minimal interconnect structures are used in between these layers: Monolithic Inter-Layer vias (MIVs) for vertical connections and very limited metal layers for horizontal connectivity. The dimension and parasitics of MIVs are of the order of a local via (in the range of nm). As a result, ultra fine-grained vertical integration of devices is possible. The integration grain is finer from the one of the 3D-SICs, since we can stack at lower level (transistor and gate).

If the stacking is happening at the gate-level, the appropriate EDA tools should perform system partitioning at gate-level, and they should be able to place & route the design in 3D. Note that current EDA tools are not ready for this, since the IC design was a 2D problem until today. If stacking happens at transistor level, the existing EDA, 2D, tools can be used because the problem is now moved to the one of the 3D standard cell design.

While 3D monolithic integration is very promising, there are lot of issues that remain associated with the efficient wafer manufacturing. 3D monolithic process requires high-temperature operations that heavily impact the device performance. Thus, it is known that two consecutive layers will not have the same performance; the top layer having worse performance than the bottom layer, since it is processed afterwards. This will have important consequences on system design and will lower the benefits of such integration.

In the context of this work we focus on 3D-SIC circuits approach.

Implementation

In order to assess physical parameters of an Integrated Circuit (IC) we need to create an IC layout model. In traditional IC design this model is typically generated after gate-level synthesis and place&route (P&R). Both steps are performed using Electronic Design Automation (EDA) tools, in a sequence of

many different tools. The whole processes typically requires many man and CPU cycles before final model is built. This is due to the system complexity; the accuracy of the models at different abstraction levels that we need to build; and the lack of completely automated methods and algorithms (most of the problems that any design flow is solving are NP-hard problems). To deal with 3D design we have extended the commercially available tools suite from Atrenta (now Synopsis).

The flow is depicted on Figure 1.4, and we do provide a bit more detailed explanation of each flow step.

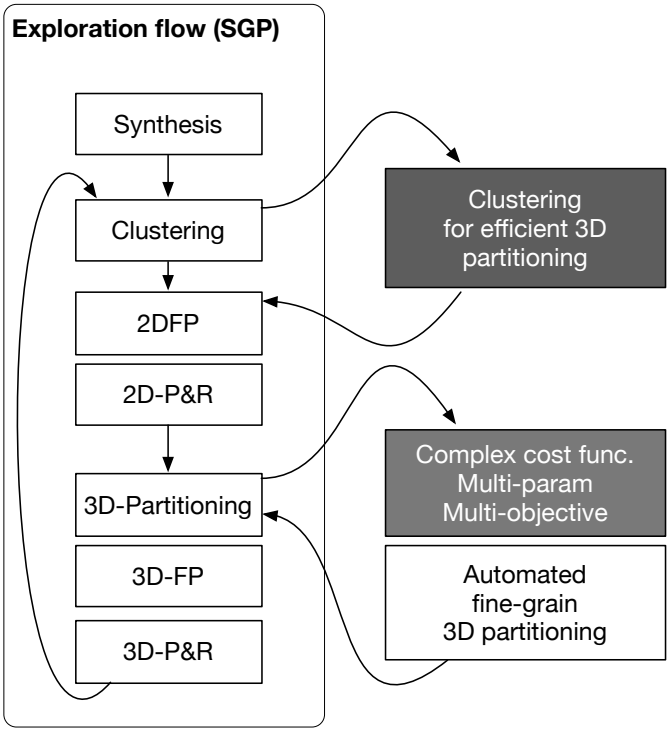

Fig. 1.4. Sequence of steps required to enable 3D-SIC design (3D Design Flow)

Step 1: RTL Synthesis – The input of the proposed flow, as in any IC design flow, consists of several elements. First we need to provide the description of the circuit itself. This is done using any of the standard Hardware Description Languages (VHDL, Verilog). Also, if present, abstracted high-level models of certain blocks cab be supplied with interface definition, area, power, timing etc. (i.e. black-boxes). Finally, typical system constraints (timing) are supplied through standardized file format (.sdc files).

Once the design database is ready, the RTL is synthesized to gate-level netlist using standard technology files (.lib and .lef files) provided by the technology vendor. Note that if the design is intended for 2.5D/3D integration, the technology files will have to capture both electrical and geometrical information not only for standard cells, but also for 3D specific features such as TSVs, micro-bumps, Cu-pads, RDLs, etc. that will be instantiated depending on the stack configuration (F2F or F2B) chosen.

Synthesized gate-level netlist is analyzed for area, timing, logical congestion and other properties. The constraints and synthesis tool guides are adapted to reach realistic goals. Once the synthesis flow is stable i.e. the synthesis process produces the netlist in line with timing constraints, generated netlist can be used as input for both 2D and 3D physical design flows.

Step 2: Clustering – Prior to floorplanning, the gate-level netlist is partitioned into a number of "reasonably" sized physical clusters of standard cells. The clustering is done so that the floorplan engine works on a reduced number of placeble instances. The size of clusters (the meaning of "reasonable") will depend on the total circuit size. In general it is chosen in such a way, so that the total number of clusters is in order of few dozens to few hundreds of physical (thus placeble) entities. This is the optimal number of instances (tool run-time vs. quality of the solution) for the floorplan engine.

The purpose of this step is twofold. First, the system is viewed as an assembly of blocks, rather then a collection of logic gates (it could be few dozen of millions) in the design to enable floorplanning (i.e. block level placement) as explained. But secondly, such system view will help in establishing what should go where in the stack.

Standard cell clustering can be performed using many different methods. Note that this is a very important step since the quality of the physical design will depend on the clustering scheme adopted. In the current design flow we can use either top-down approach (from top-level of the design, way to the standard cell level) or bottom-up (from the standard cell level and up). Different clustering objectives could be achieved during clustering: keeping and following the logical hierarchy, creating clusters of the similar size, hierarchical min-cut across the clusters, etc. Note that in the case of the 3D integration, the clustered netlist is further partitioned into a number of gate-level entities (that will remain clustered), equal to the number of dies in the system (see next step, 3D-Partitioning).

Step 3: 3D-Partitioning – This step is performed only in a 3D flow. The stack structure: the number of dies, technology node on per die basis

(this is to support heterogeneous integration), stacking orientation (face-up or face-down), 3D structure properties (TSV/ubump/CuPad) and RDL net properties (width/pitch), etc. are specified in a manually generated XML file, given as an input to the tool. The actual 3D partitioning of the gate level netlist is carried out in an automated fashion, using the stack configuration file, synthesized gate-level netlist.

For 2.5 and 3D designs, the synthesized gate-level netlist is partitioned into so many gate-level entities as there are dies in the system. Depending on the 3D integration scheme appropriate inter-die interconnect models are applied. To enable the partitioning, the designer first needs to specify the initial stack structure. This is done with a manually created XML file. The stack is divided into tiers, each tier can contain multiple dies (all dies in the same tier have the same z coordinate). XML specifies also the orientation of each die in the stack (face-up or face-down). Each die container can also define its own TSV, micro-bumps, RDL properties that will override those specified in the technology file. This is to insure that during design exploration phase we can easily replace one basic technology parameter and understand the impact on the system performance (e.g. TSV size, form factor/pitch).

- User specified partitioning directives and
- Automated partitioning

In the case of user specified partitioning directives, the information on which block should go where is provided manually in the form of explicit block-to-die assignment directives using a dedicated tool command. However in of fine pitch 3D interconnects we could perform fine grain partitioning, i.e. system blocks are smaller and smaller. Thus their number, as well as their interconnectivity view, will increase considerably making manual partitioning process impossible.

In order to automate the partitioning we use graph theory, already extensively used in the field of the VLSI design. Graph structure, with vertex and edge mimic perfectly well a logic circuit, no meter the level of hierarchy we are looking at (although it might become very complex as we go down in the logical hierarchy). Graph vertex represents a logic gate or a cluster of standard cells (whatever the size of that cluster in terms of gates might be). The edge models the connection(s) between the gates (or clusters).

One of the particular problems that have been extensively covered in the graph theory literature concerns graph partitioning problem. For this problem the algorithm tries to automatically produce two, or eventually more graph partitions that have specific properties. Most of the time these properties aim certain cut objective: like min-cut, in which the sum of the weights of the cut edges is minimal. This can be eventually combined with the objective on vertexes that could be equally balanced between the partitions.

Graph partitioning is illustrated in Figure 12 where we show a simple graph with both edges and vertexes being weighted (the numbers between the square brackets). The initial graph, shown on the left is partitioned into

two partitions, providing a min-cut on the edges (cut cost of 5) and balanced vertex weights (respectively the cost of 12 and 11 for Die0 and Die1).

Once the partitioning information is generated (manually or automatically) the design is effectively partitioned. The tool performs module assignment for a given tier. During this process all inter die nets will be automatically extracted and corresponding physical inter-die net models applied depending on the die orientations (choice between F2F and F2B) and specified technology options. With internally partitioned netlist we can now proceed with floorplanning of the design, and this is then followed by standard cell placement and routing. At this stage the approximated layout of the circuit is generated and it can be characterized to assess system performance. Typically we extract area, congestion and timing analysis, power dissipation per component etc.

Step 4: Floorplanning – The clustering and 3D partitioning steps are followed by the floorplanning step. In case of 2D, the floorplanning is carried out automatically, with physical constraints that are manually generated based on connectivity analysis and whatever knowledge/constraints we might have on the design (e.g. hard-macro pre-placements etc.). In case of 3D, additional physical constraints related to 3D net placement are generated (TSV/micro-bump clustering and placement). Floorplanning is carried out for each die separately. Becasue

Step 5: Standard cell Placement and Routing (PNR) – Standard cell placement and routing are carried out after floorplanning. In case of 3D, it is performed separately for each die, in a sequential fashion.

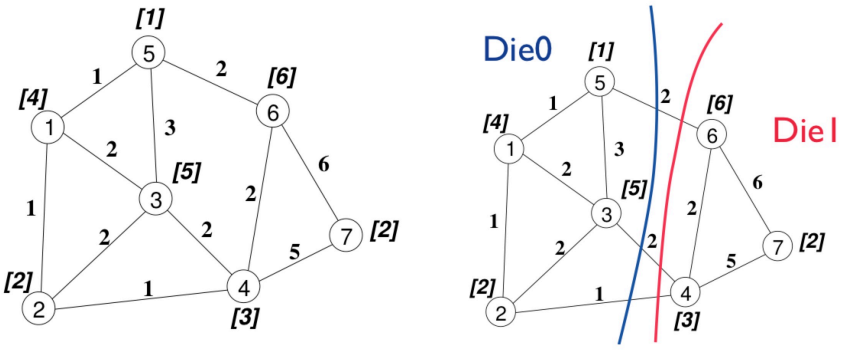

Fig. 1.5. Simple graph with annotated vertices (system blocks and their area) and edges ()

1.3 Case study

1.3.1 Experimental set-up

Design and implementation

We have carried out experiments using three different sub-blocks of the OpenSPARC-T2 SoC design: the core (SPC) that is gate dominated; the crossbar circuit (CCX) that is wire dominated; and the Ethernet module (RTX), an example of "typical" circuit. For graph formation hyper-edges have 10 different cost functions: Min-cut based (number of wires per C2C connection, average wirelength of all nets for each C2C connection, total WL of all nets in each C2C connection, product of 1 and 2, product of 1 and 3) and the inverse of the above. For clustering we have considered logical (four different hierarchical levels), top-down (two sizes) and newly developed bottom-up clustering methods (two sizes). For graph vertices weights we have been considering Area (as in the past) and area & power.

Each design results in 80 points (8 clustering schemes X 10 cost functions per hyper-edge = 80 runs) that we analyze for key distribution parameters shown as box-plot (indicating Max/Med/Min and spread) for total and maximum wirelength. The following observations can be made based on the results obtained and shown on Figure a) below. a) Wire dominated circuits (crossbar) benefit more from 3D then the others (gains are higher then usual 50

Because the choice of cost functions (clustering) is not correlated with the gains across different designs, all combinations of clustering/hyper-edge costs are considered (here 80). From the design flow perspective this is not prohibiting, because the flow is fast (1h for the above). To deal with such a huge design space we have implemented a method to choose Pareto dominant points only. To illustrate the benefits we have looked into the crossbar circuit and the initial search space of 80 solutions. If we consider simultaneously the following optimization objectives: minimum total wirelength, minimum critical path wirelength and maximum of 3D nets, by using the proposed solution we can reduce the space to 19 points that can be then handled manually.

To better understand the impact of the 3D structure pitch we have compared: the total wirelength and the maximum wirelength as function of Number of 3D nets. This is shown on Figure b)(normalized to max Nº of 3D nets). Clearly a higher density of 3D nets will allow: less total wirelength, hence less congestion/power; finally less maximum wirelength means better performance. However the total and maximum benefits are converging. Finer grain partitioning should thus optimize other criteria.

As for power-aware partitioning, two problems had to be solved upfront. First, hyper-graph partitioning can handle only one weight per vertex/edge and linear combination of power/area will not work in this case. To solve this we have adopted normal graphs, since they can make use of more then 2 weights per graph vertex. Secondly, the graph partitioning (hyper and normal) can't produce unbalanced partitions (e.g. max vertex weight in 1st and

min in 2nd partition). To solve this problem we have introduced the following solution. A "Dummy node" is introduced in the graph. This node has (almost) zero area & power that offsets the power budget to a certain value. This "Dummy node" is pre-assigned to the low-power die and will thus force unbalanced partition. To illustrate the impact of the power aware partitioning on other system parameters we have applied the proposed framework to OpenSPARC core shown on Figure c). With targeted power unbalance of 70-30

3D system partitioning

1.4 Case study

1.5 Results

1.6 Conclusion and future works

References

[1] S.F. Al-Sarawi, D. Abbott, and P.D. Franzon. A review of 3-d packaging technology. *Components, Packaging, and Manufacturing Technology, Part B: Advanced Packaging, IEEE Transactions on*, 21(1):2 –14, feb 1998.

[2] E. Beyne and B. Swinnen. 3D System Integration Technologies. *Integrated Circuit Design and Technology, 2007. ICICDT '07. IEEE International Conference on*, pages 1–3, May 2007.

[3] S. Borkar. Design perspectives on 22nm cmos and beyond. In *Design Automation Conference, 2009. DAC '09. 46th ACM/IEEE*, pages 93–94, July 2009.

[4] Robert X. Cringely. Breaking moore's law @ONLINE, 2013.

[5] Yi Cui, Zhaohui Zhong, Deli Wang, Wayne U. Wang, and Charles M. Lieber. High performance silicon nanowire field effect transistors. *Nano Letters*, 3(2):149–152, 2003.

[6] Shamik Das, Andy Fan, Kuan-Neng Chen, et al. Technology, performance, and computer-aided design of three-dimensional integrated circuits. pages 108–115, 2004.

[7] J. Figueira, S. Greco, and M. Ehrgott. *Multiple criteria decision analysis: state of the art surveys*, volume 78. Springer Verlag, 2005.

[8] D. Goldhaber-Gordon, Hadas Shtrikman, D. Mahalu, David Abusch-Magder, U. Meirav, and M. A. Kastner. Kondo effect in a single-electron transistor. *Nature*, 391(6663):156–159, January 1998.

[9] Quantin Hayez, Yves De Smet, and Jimmy Bonney. D-Sight: A New Decision Making Software to Address Multi-Criteria Problems. *IJDSST*, 4(4):1–23, 2012.

[10] Bertrand Mareschal and Jean-Pierre Brans. Geometrical representations for MCDA. *European Journal of Operational Research*, 34(1):69–77, February 1988.

[11] Dragomir Milojevic, Ravi Varadarajan, Dirk Seynhaeve, and Pol Marchal. *PathFinding and TechTuning in Three Dimensional System Integration*. Springer, Nov 2011.

[12] R.S. Patti. Three-dimensional integrated circuits and the future of system-on-chip designs. *Proceedings of the IEEE*, 94(6):1214–1224, June 2006.

[13] S.J. Tans, A.R.M. Verschueren, and C. Dekker. Room-temperature transistor based on a single carbon nanotube. *Nature*, 393(6680):49–52, 1998.

[14] V.V. Zhirnov, III Cavin, R.K., J.A. Hutchby, and G.I. Bourianoff. Limits to binary logic switch scaling - a gedanken model. *Proceedings of the IEEE*, 91(11):1934 – 1939, nov 2003.