Computationally Efficient Standard-Cell FEM-based Thermal Analysis

DOI:
10.1109/ICCAD.2017.8203817

Document Version
Accepted author manuscript

Computationally Efficient Standard-Cell FEM-based Thermal Analysis

ABSTRACT

Thermal analysis is a high performance computing problem because the microscale spatial and time discretization of the heat equation translates into repeatedly solving large linear systems of equations. In previous works, compact models of integrated circuits (IC) were introduced to speed up this process. However, such methods are limited in their accuracy as they approximate the underlying physics. The solution methodologies for such models are also ill-suited to simulate the thermal characteristics of an IC at cell-level. The finite element method (FEM) is an appropriate computational technique for providing both fast and accurate thermal analyses. Considering that the number of cells in modern ICs is in the order of millions, thermal analysis at this abstraction level is a formidable task. Consequently, handling the computational meshes and computing thermal profiles of an IC at the cell-level requires intensive computing power. In order to provide accurate cell-level thermal simulations at a lower computational cost, this work introduces an advanced mesh generator for cell-level floorplans. This mesh generator applies a cell-homogenization technique based on the initial cell-level floorplan and the power trace to create reduced order meshes for efficient cell-level thermal simulations. The effects of using different homogenization algorithms are explored to illustrate the tradeoff between the simulation speed and the solution accuracy. Results show that the proposed technique can achieve a 90% reduction in the number of nodes in the mesh with less than 5% error to the temperature of the full scale mesh. In addition, the simulation time is reduced by an order of magnitude.

1. INTRODUCTION

Modern CMOS technologies suffer from thermal issues due to self-heating, high power densities, and the low thermal conductivity of their materials [1, 2, 3]. These problems lead to performance degradation and excessive leakage power that can hinder higher integration densities [4]. Emerging 3-D integration technologies are promising to keep performance scaling as the post device-scaling era is reached. However, these technologies also exhibit thermal issues since the multi-tier active layers induce high power densities in a package without a sufficient number of heat dissipation pathways [1, 5, 6].

Thermal analysis tools that simulate and predict such thermal issues are crucial and many academic thermal simulators for ICs have been introduced [7, 8, 9, 10, 11]. In general, these tools take as input a floorplan of the functional blocks of the IC and convert it to a computational mesh, i.e., subdivide the IC into smaller tetra or hexahedral (cuboid) elements for simulations. These tools are suitable to early stage design specification and validation. However, existing tools are impractical for post-*Automatic Place and Route* (APR) thermal simulations. Considering the role of temperature in the reliability and aging mechanisms of circuits, post-APR simulations are especially important [12, 13, 14]. Existing tools use relatively large (cuboid) elements to describe the physical implementation of the circuit, which fundamentally limits the granularity of the analysis. As a result, these tools do not allow for sufficiently accurate analysis within reasonable computational times. In addition, cells known to potentially become hot spots (e.g., clock buffers) cannot be individually analyzed.

Moreover, the computational methodologies used in these tools are ill-suited to solve the large linear systems generated from the discretization of a post-APR floorplan. The narrow flexibility to simulate the complex geometries of an advanced IC is also a significant limitation. In order to avoid these limitations, other methods, such as the finite element method (FEM) can be used to solve the heat equation [15]. The FEM discretizes the heat equation directly, avoiding the accuracy pitfalls inherent in compact models. The sequence of resulting linear systems can then be solved with fast preconditioned iterative solvers [8, 15, 16]. Furthermore, computational methodologies that use the FEM can model more versatile geometries, while accurately and efficiently solving the post-APR heat problem when compared to compact models.

Simulating post-APR thermal characteristics by the FEM requires a computational mesh of the IC, which is generated from its *cell-level* floorplan. The number of vertices or nodes in theses meshes, commonly referred to as degrees of freedom (DOF), is typically several orders of magnitude larger than the number of cells in the IC. This behavior can severely restrict the number of cells that can be analyzed efficiently. For example, using standard finite element libraries, a machine with 32 GB memory can generally handle problem sizes on the order of 10 million DOFs. However, cell-level floorplans can easily generate meshes with hundreds (even thousands) of millions of DOFs, making such computations infeasible without massive parallel computing power. To reduce the number of DOFs, this paper introduces a novel mesh generator that creates a computational mesh from a cell-level floorplan while applying an advanced cell-level homogenization algorithm to curtail the number of DOFs in the problem without sacrificing the accuracy of the thermal

profile.

The specific contributions of this paper to the problem of cell-level thermal analysis for an IC are:

- the ability to convert the industrial standard design exchange format (DEF) and library exchange format (LEF) files to a computational mesh, which also supports future vertical integration technologies,
- an advanced cell-level homogenization (merging) technique to effectively lower the number of DOFs in the mesh,
- a tunable homogenization algorithm based on spatial and power information,
- a standalone framework that can be easily integrated with the commercial EDA exchangeable files, such as DEF and LEF, and FEM-based thermal analyzers down to the standard cell-level.

The rest of the paper is organized as follows. Section 2 reviews the FEM and discusses the motivation for the proposed mesh generator. Section 3 introduces the mesh generator which requires as input the industrial exchangeable format DEF and LEF files. Section 4 presents the details of the cell homogenization algorithms. Section 5 presents the results of the numerical experiments which illustrate the advantages of the proposed approach. Section 6 offers some concluding remarks.

2. BACKGROUND AND MOTIVATION

The temperature $u(\mathbf{x},t)$ (in [K]) at point $\mathbf{x}=(x,y,z)$ and time t in an IC is governed by the following initial boundary value problem

$$\rho\, C\frac{\partial u(\mathbf{x},t)}{\partial t} - \nabla\cdot(\kappa\nabla u(\mathbf{x},t)) = f(\mathbf{x},t) \text{ in } \Omega\times[0,T], \quad (1a)$$

$$\kappa\nabla u\cdot\mathbf{n} = \eta(u_a - u) \text{ on } \partial\Omega\times[0,T], \quad (1b)$$

$$u(\mathbf{x},0) = u_a. \quad (1c)$$

The physical domain of the circuit is denoted by Ω and its boundary by $\partial\Omega$. The physical parameters are the material density ρ $[kg/m^3]$, specific heat C $[J/kgK]$, thermal conductivity κ $[W/mK]$, and thermal transmissivity η $[W/m^2K]$ at the boundary. Note that $c_V = \rho C$ is the volumetric specific heat. The function $f(\mathbf{x},t)$ $[W/m^3]$ is the power density dissipated by the active layer(s) of the system. The Robin boundary condition described by (1b) represents Newton's law of cooling, i.e., the heat flux at the boundary is proportional (with the constant heat transfer coefficient η) to the difference between the ambient temperature u_a and the temperature at the wall.

At the core of the thermal analysis process is the discretization and numerical solution of (1). In contrast to previous approaches that invoke the electrothermal duality to derive compact models to discretize the heat equation, such as [9, 10], the finite element method (FEM) is used here [15]. The advantage of the FEM in this case is the ability to more accurately represent the underlying physics of the problem, i.e., the heat equation is directly discretized, rather than computing an approximation of the equation. In addition, the FEM easily supports the discretization of more general geometries and handles the implementation of various boundary conditions.

The FEM first partitions the domain into a union of non-overlapping elements $\Omega = \cup_k \Omega_k$. The elements Ω_k are typically tetra or hexahedra (cuboid) with a characteristic length h called the mesh width. The discretization of the domain into the finite elements is the mesh generation process. The finite element solution is obtained by interpolating the computed discrete nodal values at the vertices of the elements, i.e., the solution has the form $u_h(\mathbf{x},t) = \sum_{j=1}^{n} u_j(t)\phi_j(\mathbf{x})$, where $u_j(t) \approx u(\mathbf{x}_j,t)$ are the n unknown nodal values, and $\phi_j(\mathbf{x})$ is a global Lagrangian basis function that equals 1 at node $\mathbf{x}_j$ and zero at every other node. The n unknown nodal values $\{u_j\}_{j=1}^{n}$, called DOF, can be collected into a column vector $\boldsymbol{u}_h = [u_1,\ldots,u_n]^T$ and this vector of unknowns is determined by solving a linear system of n equations. Finely resolved computational meshes with many nodes translate into large (sparse) linear systems. Despite fast solution methods, working with such fine meshes and large problem sizes results in prohibitive computational costs. Therefore, reducing the order of the meshes and maintaining the accuracy of the original finer (larger) meshes is an essential feature for thermal analysis.

2.1 Structured Meshes

The proposed mesh generator creates structured grids consisting of hexahedral (cuboid) elements. Structured grids generally consist of hexahedral elements (though tetrahedrons are also possible) that follow a repeatable pattern; see Fig. 1. There are several advantages to using a structured grid over an unstructured (non-repeatable) grid. First, due to the regular pattern, the underlying data structures require less memory and make refinement of the mesh (uniform or adaptive) much faster. In addition, such structures result in regular sparsity patterns in coefficient matrices which allow for better caching and faster execution [17]. Lastly, the regular cuboidal structures of integrated circuits make such meshes a natural choice to use.

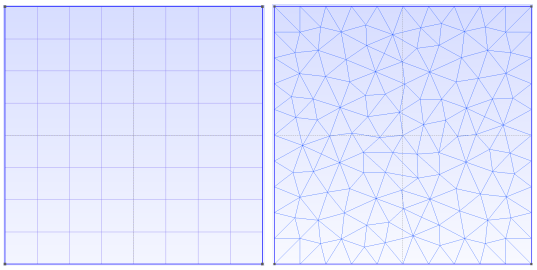


Figure 1: Depiction of a structured and unstructured mesh

Ensuring quality of the mesh is extremely important and the proposed mesh generator offers a number of pre-simulation refinement routines for achieving this. Certain circuit designs may produce meshes with stretched elements and reentrant corners at overhanging boundary surfaces such as heat sinks. Without additional local mesh refinement, the initially generated meshes for these structures can produce solutions with lower accuracy, e.g., unphysical damping of the temperature. As a result, the proposed mesh generator handles the surroundings of a circuit by supporting local mesh refinement of these structures.

3. MESH GENERATOR

This section introduces the proposed mesh generator that converts a cell-level floorplan described in a DEF and corresponding LEF file to a structured mesh. The details of this process, which includes the merging of the DEF and LEF files into a single XML file containing the coordinates, sizes, and physical parameters of the cells, and its conversion to the structured mesh by parsing the XML file, are described in Section 3.1 and 3.2, respectively. A number of specific definitions are used in the description of the mesh conversion process and are listed in Tab. 1.

Placed cells	standard placed cells defined in the DEF and LEF files
XML-element	a component definition in the XML file hierarchy
Atomic cells	the quadrilaterals defined by the grid space

Table 1: Definitions of terms.

3.1 DEF-LEF to XML Conversion

In modern CMOS design flows, post-APR design information, such as the physical layout of cells, is defined in a DEF file. The DEF stores the anchor locations of the cells and the associated cell models in a standard cell library. However, the size information of the cells is not contained in this file. This information is defined in an LEF file provided by foundries. Creating a computational mesh of an IC with complete geometric information of the cell on the die therefore requires both files.

The proposed mesh generator is designed as a two stage operation. First, the XML file is created and then the mesh is generated. The reason is that the DEF and LEF files only contain spatial information of the circuit, which is insufficient for a thermal simulation. This situation is due to the fact that the DEF and LEF files have limited backend information of the circuit. Thermal simulations that accurately capture the thermal profile require a full geometric definition of the IC, along with its substrate, package, and heat sink. Therefore, the step of creating the XML file from the DEF and LEF files is just an intermediate step which requires manually defining the other components of the circuit and its surrounding package so that the IC and package is modelled.

In Fig. 2, an abstract hierarchical view of a single tier IC in an XML file is shown. The components highlighted

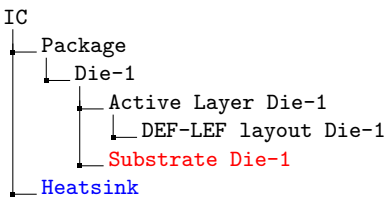


Figure 2: XML Hierarchy of a one tier IC

in black can be generated from a DEF and LEF file. However, the missing substrate definition highlighted in red is required before creating the XML file for the package and its components. A stack description file (.stk) is introduced that defines auxiliary information of a die including the thickness of its substrate, a DEF file, the corresponding LEF file, and the anchor point for the cells in the DEF and LEF files.

The mesh generator creates the hierarchy of the package in the XML file based on the stack file. The stack file also supports the definition of multiple dies, which can be used to describe multi-tier 2.5-D and 3-D IC technologies. Other components, e.g., heatsinks (highlighted in blue), can be manually integrated in the XML file afterwards. The hierarchical structure of the XML file allows the package to be described via sub XML-elements. Intermediate XML-elements between the package and Die-1 can also be defined. The XML file is then parsed in order to generate a computational mesh. The mesh generator adheres to this hierarchy of the XML file during the creation of the components of the subdomains.

Standard cell information is extracted from the LEF file by using the API provided by Si2 [18]. The mesh generator reads in geometric information such as the width, length, and height of the backend of the fabrication process, which can be applied to represent the height of standard cells with the negligible height of the frontend in the substrate. It also supports a user-defined bypassing list that bypasses standard cells of fillers or other types of standard cells that do not contribute power to the IC. This bypassing list helps reduce the number of DOFs and is further discussed in Section 4. Then, the mesh generator creates XML elements for the placed cells defined in the DEF file by using the Si2 API [18]. These elements are subdomain definitions of a default layer referred to as the *active layer*, which is a rectangular cuboid defined by the die area and the height of the standard cells.

In order to create full information of the package, the mesh generator also produces a substrate layer defined in the stack file at the same level in the hierarchy as the *active layer* and then wraps these definitions under the package. When the XML elements are produced, the mesh generator also inserts the default physical parameters to each XML element. Heatsinks must be manually defined after creating the basic XML file from the DEF and LEF files. Once created, the XML file can be easily modified to include additional layers or elements in the subdomain. The mesh generator supports heatsinks as defined in Appendix A.

3.2 XML to a Structured Mesh Conversion

Once the XML file contains the full geometry of the IC required for the thermal simulation, the mesh generator parses the XML file and produces the computational mesh. As mentioned in Section 2, the mesh generator provides a structured mesh. The structured mesh is output in a .msh file using the mesh format as in [19]. Algorithm 1 summarizes the operation of converting the XML to a structured mesh.

To create a structured mesh, the mesh generator reads the XML elements in the hierarchy of the XML file with the breadth-first search algorithm and sequentially stores the XML elements in a container (vector). The algorithm loops through this container and records the near and rear (x, y, z)-coordinates of all XML elements into separate grid containers x-grid, y-grid, and z-grid. After sorting and removing duplicates in these grid containers, the coordinates of these sorted grid containers become grid lines to build the structured mesh in the x-, y-, and z-directions.

These grid containers describe a grid space containing

Algorithm 1 XML to a Structured Mesh

```
1: BFS: breadth-first searching
2: procedure MESH(.xml, .msh)
3:     hierarchy ← hierarchy of elements in .xml
4:     elements_list ← BFS(hierarchy)
5:     for all elements in elements_list do
6:         XYZ_grid.push_back(xyz.near/rear)
7:     end for
8:     Sort and Unique(XYZ_grid)
9:     create grid_space from XYZ_grid
10:    for all elements in elements_list ▷ (reversely) do
11:        sub_space ← xyz.near/rear in grid_space
12:        for all atomic_cells in sub_space do
13:            if atomic_cells.label = undefined then
14:                atomic_cells.label ← element.PhysicalID
15:            end if
16:        end for
17:    end for
18:    for all atomic_cells in grid_space do
19:        if atomic_cells.label = true then
20:            nodes.push_back(atomic_cells.nodes)
21:            if atomic_cells.faces is at boundary then
22:                faces.push_back(atomic_cells.faces)
23:            end if
24:            volumes.push_back(atomic_cells.volume)
25:        end if
26:    end for
27:    .msh.push_back(nodes)
28:    .msh.push_back(faces)
29:    .msh.push_back(volumes)
30: end procedure
```

the atomic cells that are not yet attached to the ID of a placed cell. The ID corresponds to physical parameters used in the thermal simulation, for instance, the thermal conductivity and the specific heat. Each XML-element of a placed cell may contain multiple atomic cells. The mesh generator loops through the XML element container again in a backward order to attach the ID to the atomic cells. Since the backward loop always reads a lower-level XML element first, the sub components of an XML element are always defined first. The mesh generator only applies an ID to an atomic cell once. This first-read and first-label mechanism guarantees the structured mesh is created according to the hierarchy of the XML file.

As mentioned in Section 3.1, the bypassing list excludes certain placed cells such as inactive filler cells.The grid lines associated to these placed cells do not need to be defined in the XML file. As a result of excluding these placed cells, fewer grid lines are generated resulting in a reduced mesh size. Furthermore, this bypassing process also reduces the searching and mapping operations of these inactive placed cells during the construction of the mesh because the undefined regions are automatically assigned to the ID of its parent element, i.e., the *active layer*.

4. CELL-LEVEL HOMOGENIZATION

This section describes the implementation details of the proposed cell-level homogenization technique for computational meshes. The mechanism of the homogenization is based on a concurrent splitting and merging of placed cells, which aggressively decreases the number of DOFs while maintaining a high level of accuracy for thermal simulations by preserving the major characteristics of the placed cells.

As previously mentioned in Section 3.1, the mesh generator first creates a grid space and then labels atomic cells with IDs in a structured mesh. The homogenization of placed cells with the mesh generator translates into removing a portion of these grid lines. The changing of the grid space can merge some atomic cells resulting in additional splitting and merging of the placed cells.

Modern CMOS technologies can be viewed as a stack consisting of multiple layers of materials in the z-direction. The device and backend layers are only a small portion of the stack and are modelled as a thin plane layer in thermal simulations. The layer is defined with as a sub XML-element of placed cells. According to the design rules of a semi-automatic flow, placed cells are located on multiple parallel rows, flanked by power rails, i.e., GND and VDD. The placed cells have the same height and varying width according to their function and driving strength. The above features of the semi-automatic flow suggest the mechanism for grid elimination needs to be performed only in the x-direction (or the direction along which cells are aligned), otherwise it will significantly affect the accuracy of thermal simulations by removing grid lines in the y- and z-directions. The following sections introduce a grid line removal scheme, based on a particular weighting of the grid lines and the power density evaluation. In addition, two hybrid techniques are proposed which combine both of these schemes. Lastly, computing the power density of merged cells is discussed.

4.1 Weighted Grid Lines

Maintaining information of the placed cells as much as possible is the top priority when generating the structured mesh. As previously discussed in Section 3.2, each grid line is defined at least once by a placed cell or by the components of the package. Here, a scheme based on weighting the grid lines in the x-direction is presented; see Algorithm 2 for the pseudocode. The method records the number of times each grid line is used by the elements defined in the XML file and then uses this information to eliminate grid lines in the x-direction to decrease the number of DOFs.

To implement this scheme, the mesh generator first creates a vector of the same size as the number of grid lines in the x-direction to record the grid line weights. These weights are defined as the number of boundary edges of placed cells attached to each grid line. During the reverse loop over the XML element list when the atomic cells are labeled with their IDs, each XML element adds one point to its corresponding near and rear x-coordinate. Grid lines are removed whenever the weight is less than a user-supplied threshold. Setting this threshold depends heavily on the characteristics of the design, such as the number and size of the placed cells. Setting a large threshold value results in a coarser mesh.

Before the mesh file is output, the mesh generator instantiates the atomic cells in the old grid space to check if the near and rear x-coordinates are valid with the new grid lines in the x-direction. This process entails first finding a valid near x-coordinate and then advancing to the next valid x-grid line for the rear x-coordinate. After finishing the search of the near and rear x-coordinates, the atomic cell is written to the mesh file. If the near and rear x-coordinates are not adjacent to the old grid lines in the x-direction, the mesh generator records the distance between the grid lines and the old IDs of the atomic cells and then labels the new atomic cell with a new ID. The distance information is used to interpolate the power trace and is discussed further in Section 4.4.

Algorithm 2 Weight Grid Lines Homogenization

```
1: procedure OUTPUT(.msh)
2:     for all z in old_z-grid do
3:         for all y in old_y-grid do
4:             for all x in old_x-grid do
5:                 shift ← 0
6:                 if x is valid then
7:                     for all S = x + 1 to old_x-grid.end do
8:                         if S is valid then
9:                             shift ← S − x − 1
10:                            break
11:                        end if
12:                    end for
13:                else
14:                    continue
15:                end if
16:                if shift ≠ 0 then
17:                    for all x to x + shift do
18:                        record(Atomic Cell ID)
19:                        record(distance)
20:                    end for
21:                    Create a new ID
22:                    Output(Atomic Cell x to x + shift + 1)
23:                else
24:                    Output(Atomic Cell x to x + 1)
25:                end if
26:            end for
27:        end for
28:    end for
29: end procedure
```

Figs. 3 to 5 show the steps of the proposed scheme on a pseudo circuit. Active cells alongside fillers of a die in a 2-D DEF layout are illustrated in Fig. 3. Converting the die to a structured mesh first requires the grid space with atomic cells for the die which are illustrated by grey dashed lines of Fig. 4. The grid lines are labelled with ID numbers at the top of the figure. The weight of each grid line is denoted by the grey number at the bottom of the figure. In this example, the objective is to remove grid lines with a weight lower than 4 in order to construct a new grid space and atomic cells for the cell-level homogenization. The grid lines of negative numbers are exempted from the elimination because they are at the boundary of the mesh. Fig. 5 shows the new grid space and atomic cells. The DOFs are reduced from 84 to 54 a reduction by 35%, which roughly translates to a 35% decrease in the simulation time.

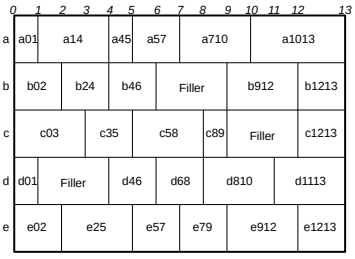


Figure 3: Cells of the die as described in the DEF and LEF files.

As mentioned in Section 3.1, the filler cells are bypassed during the conversion of the DEF-LEF to XML step. The grey numbers shown at the bottom of the figure are the weights of the grid lines which enclose filler cells. If the threshold is set to 4, then grid lines *1* and *6* are included in the new grid space which yields 66 DOFs. However, some atomic cells, such as the filler cells at row d and grid lines *1* and *2*, contribute none or minimal energy to the thermal

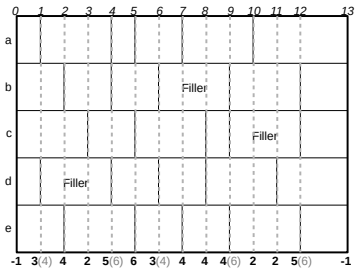


Figure 4: Grid space for conversion to a structured mesh.

simulation. Thus, these atomic cells are ideal candidates to be merged. The most efficient way to deal with these atomic cells is through the bypassing procedure introduced at the DEF-LEF to XML step.

In Fig. 5, observe that some boundaries, shown in grey dash lines, of the placed cells are removed. This removal indicates that the placed cells related to these boundaries are either merged or split. For example, the placed cell *a14* in Fig. 3 is split into left and right parts. The left part is merged with *a01* to form a new placed cell (i.e., *a02*). The updated mesh is illustrated in Fig. 6, where the newly created cells are highlighted in grey and underline. The linked power density information of the atomic cells is also shown in Fig. 6 by the IDs of the placed cells. The cells highlighted in grey and underline require new power traces which are generated from the existing power trace. The technique to modify the power trace to match the new grid is discussed in Section 4.4.

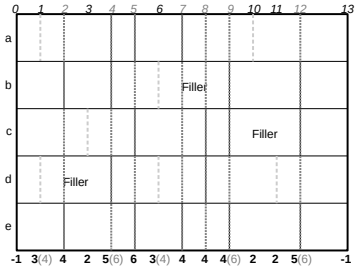


Figure 5: New grid space for merged atomic cells.

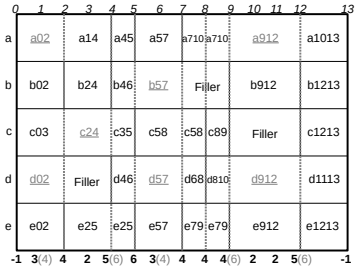


Figure 6: Final mesh with newly merged/split placed cells.

4.2 Power Density Evaluation

Another scheme to remove grid lines along the x-direction is based solely on the power trace information. This scheme helps identify which grid lines contribute significant power to thermal simulations in order to determine whether a grid

line is eliminated or not. The power density scheme helps to preserve the important characteristics of the thermal profile.

The implementation details are similar to the scheme of the weighted grid lines. A vector of the same size as the number of grid lines in the x-direction is created to record the value of the power density weight per grid line. Each atomic cell has an associated power density from the placed cell. The power density weight is defined as the sum of the power densities contributed by each atomic cell attached to the grid line. The mesh generator reads in a file of average power for all active components, which can be obtained from standard EDA tools, such as PRIMETIME PX [20]. When the atomic cells are labelled with their physical IDs in the grid space, the mesh generator converts the average power of the placed cell to the average power density and then associates this value to each grid line in the x-direction. These power density values are summed for each grid line and inserted at the appropriate location in the power density vector. Each entry in the vector represents an evaluation of the significance, in terms of power, of each grid line in the x-direction.

Fig. 7 is an example of the power density evaluation during the labelling process. The placed cells *a14* and *c35* highlighted in grey and underline have a power density of 10 and 1, respectively. While labeling the ID of *a14* to its atomic cells, the process links the power density to the corresponding location in the vector. The grid line *1* and *4* contribute 10 each, but lines *2* and *3* add 20 because they are used to create atomic cells twice during the labelling process. The same mechanism is applied to the placed cell *c35*, where grid line *3* and *5* add 1 while *4* adds 2. The grey numbers at the bottom of the figure are the sum of power density for each grid line after the labelling process of *a14* and *c35*.

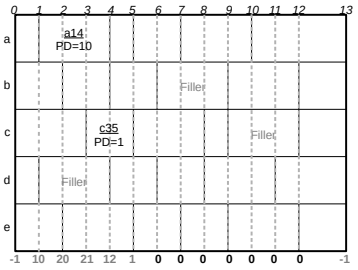


Figure 7: An example of power density evaluation in labeling process. (PD = power density)

After labelling all required atomic cells in the grid space, the vector contains the information of the sum of power density for each grid line that can be used to eliminate trivial grid lines. A pre-defined lower-bound power density threshold is applied to remove grid lines that exhibit a lower power density. This threshold value depends on the physical and power integration density of each circuit. Setting a small threshold value produces finer meshes for more accurate thermal simulations. The output algorithm of the mesh is similar to Algorithm 2 but with a vector of power density values in the x-direction.

4.3 Hybrid Modes

4.3.1 Manual Hybrid Mode

Hybrid mode is a combination of the weighted grid lines and power density approaches to eliminate grid lines. Manual hybrid mode requires the user directly input threshold values for both schemes. This operation can require a-priori knowledge of the design to better set these thresholds. It provides flexibility for specific applications, i.e., aggressively eliminating grid lines based on one scheme over the other; however this approach depends on prior knowledge of the simulated circuit.

4.3.2 Smart Hybrid Mode

Smart hybrid mode automatically generates thresholds to optimize the reduction of nodes in the mesh. The thresholds are generated based on a preset scale factor for reducing the DOFs in a mesh. For example, a scale factor 0.2 means that the mesh generator will create a new mesh that has roughly 20% of DOFs of the original, full mesh. The tradeoff between the accuracy and simulation time of the scaled meshes is reported in Section 5.1. The smart hybrid mode is similar to both schemes in that it creates vectors for both weighted grid lines and power density accumulation. However, it has two additional vectors that duplicate the original vectors of weighted grid lines and power density accumulation. It then sorts these newly created vectors from largest to smallest threshold value. The mesh generator advances both vectors simultaneously from location 0 to seek a proper threshold for both schemes. Since the proposed homogenization is based on the elimination of grid lines in the x-direction which linearly decreases the DOFs, we can use the percentage of valid grid lines in the x-direction as a stopping criterion when searching for proper thresholds in both schemes.

4.4 Power Trace

The cells that result from splitting and merging are labelled as new atomic cells in the mesh generator. For example, the grey and underline cells in Fig. 6 are newly created cells. These atomic cells have no corresponding power information required for thermal simulations. In Algorithm 2, the run-time creation of physical IDs is attached to these cells and the corresponding distance of shifted grid lines is recorded. This information is used to interpolate the power trace values which links them to the newly created atomic cells. Let $f^i = f(\mathbf{x}, t_i)$ denote the power density of a newly merged cell, then the power density of this new cell is computed as the sum of the proportional power density from the merged atomic cells. Specifically,

$$f^i = \frac{\sum_n f_n^i d_n}{\sum_n d_n}, \tag{2}$$

where the sum is taken over the n merged atomic cells and d_n is the length (in the x-direction) of atomic cell n.

Implementing this function in the thermal simulator is straightforward as this function can be implemented when adding contributions to the source vector f in (1a). The mesh generator dumps a .opt file in the ASCII format that records the IDs of the new atomic cells, the old IDs, and the distance between the grid nodes. This information can be used in a thermal simulator to generate at run-time the corresponding power density for the new atomic cells without creating a new power trace.

Accessing and altering the internal design of the thermal simulator is beyond the scope of this work. Therefore, emphasis is placed on providing a more general method

where a new power trace file containing only power information (in [W]), is generated based on the old power trace and the .opt file. Generating a new power trace value $\bar{f}^i = \bar{f}(\mathbf{x}, t_i)$ of the homogenized cell requires the volume information from the original cells and the formula for computing this value is

$$\bar{f}^i = \frac{\sum_n (\bar{f}^i_n / V_n) d_n}{\sum_n d_n w_n h_n}, \quad (3)$$

where the sum is again taken over the n merged atomic cells, V_n is the volume of atomic cell n, and w_n, h_n are the width and height of atomic cell n, respectively.

The power trace is first converted to a power density trace and is interpolated for new atomic cells. This trace is subsequently converted back to a power trace for the new atomic cells. A patch based on equation (3) will be publicly released to create a new power trace file for the mesh with the proposed cell-level homogenization.

5. RESULTS AND DISCUSSION

This section discusses the quality of the meshes generated by the proposed homogenization algorithms for cell-level thermal analysis. In addition, a design space exploration addressing the tradeoffs between the accuracy and size of the reduced order mesh. A LDPC flat design with TSMC 65 nm technology is the considered benchmark in the following experiments. The LDPC has 67K cells and is attached to a heatsink for the thermal simulations. This design is converted to a structured mesh containing 9.3M DOFs and the die consists of 3.9M DOFs. The thermal simulations were executed using an academic thermal analysis tool. The generated meshes are based on the open source GMSH [19] format which can be easily incorporated in many existing FEM-libraries. As a result, any FEM-based thermal analyzer compatible with GMSH can incorporate the proposed cell-level homogenization technique.

5.1 Quality of Meshes

To demonstrate the tradeoff between the speed and accuracy of the proposed cell-level homogenization algorithm, we compute the maximum point-wise error between coinciding nodes of the original non-merged mesh (9.3M DOFs) to a series of reduced order meshes at a fixed vertical level of the die, i.e., $\|\mathbf{u}_{ori} - \mathbf{u}_{red}\|_\infty$. The smart hybrid mode is applied to reduce the size of the meshes by 10%, 25%, 50%, and 75% for a steady state simulation. The results are presented in Tab. 2. Observe how the error increases as the number of total DOFs (including the heatsink) increases. For the full scale thermal simulation, the temperature range was 2.32 degrees. As a result, the relative error for the 10% reduced mesh is only 4.4%. In addition, the original simulation time was 466 s and the simulation time decreases to 49 s for the 10% reduced mesh, i.e., a roughly ten-fold reduction in the simulation time with only a 4.4% error.

Fig. 8 shows a histogram of the number of nodes at each temperature value. The histogram has been normalized by the total number of DOFs of the original, full scale mesh. Observe that the 75% mesh is very close to 1 after normalization which indicates that most of the characteristics of the thermal profile of the full mesh are preserved. This demonstrates that the original mesh size is efficiently reduced without sacrificing the accuracy. The

Table 2: Point-wise error comparison at $z = 0.02$ mm of the die.

	Original	75%	50%	25%	10%
DOFs	9.3 M	7.0M	4.6M	2.3M	966K
$\|\mathbf{u}_{ori} - \mathbf{u}_{red}\|_\infty$	-	0.011	0.028	0.047	0.103
Time (s)	466	407	246	123	48.8

50% and 25% reduced meshes are also close to 1 for most of the temperature values demonstrating that a 25% reduced mesh still recovers a majority of the thermal characteristics of the full mesh. On the other hand, the discrepancy of the 10% reduced mesh to the full mesh shows that setting a small scale factor can produce less accurate temperature profiles. However, based on the errors reported in Tab. 2, the point-wise error is still within 0.103 degrees.

Most of the temperature values are preserved for the 75%, 50%, and 25% reduced meshes. However, there is a discrepancy between the nodes at the higher and lower temperature values. This situation results from the application of the smart hybrid mode which preserves most grid lines near hotspots but eliminates grid lines near lower temperatuers. A discussion of the phenomenon is provided in the following section.

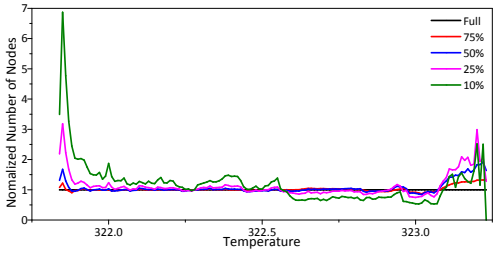


Figure 8: Normalized histogram of the number of nodes at each temperature value.

5.2 Design Space Explorations

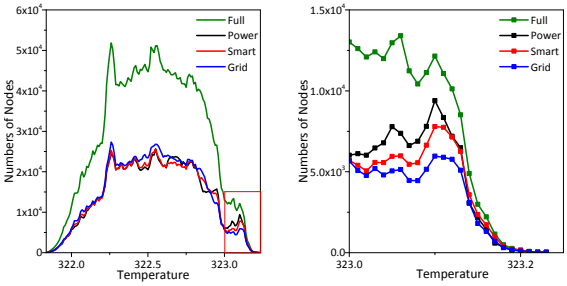
Here, the design space exploration of the proposed schemes, i.e., the weighted grid lines (Grid), power density evaluation (Power), and smart hybrid modes (Smart) is discussed. Tab. 3 shows the impact of the schemes on meshes with different levels of homogenization. The values reported in the table are the root-mean-square distances between the normalized number of nodes at each temperature value of the homogenized meshes to the original mesh.

Table 3: Root-mean-square distances to normalized number of nodes at different temperatures

	75%	50%	25%	10%
Power	1.03581	1.06389	1.18751	1.50072
Smart	1.02412	1.0825	1.19249	1.41324
Grid	0.99878	1.01275	1.006	1.01017

Observe that the weighted grid lines method is close to 1 for all reduced meshes. This indicates that it uniformly reduces the size of each mesh. This is expected behavior since the method is solely based on spatial information. Alternatively, the smart hybrid mode has a larger discrepancy away from 1, especially at higher and lower temperatures. This phenomenon is depicted in Fig. 8. The non-normalized distribution of the nodes at different temperature values is shown in Fig. 9(a) with an

enhanced view of the higher temperature values in Fig. 9(b). Note that the power density evaluation and smart hybrid mode preserve the high temperature nodes better than the weighted grid line scheme. Since it is preferable to accurately capture the thermal profile around the hotspots, it is better to consider the power density evaluation and smart hybrid mode schemes. Moreover, since the smart hybrid mode also preserves the nodes at a lower temperature better than the power density evaluation, it is recommended for homogenized cell-level thermal analysis.



(a) Histogram of the number of nodes at each temperature value (b) Enhanced view of the number of nodes at high temperature values

Figure 9: Numbers of nodes at different temperatures.

6. CONCLUSION

In this paper, a new mesh generator was introduced to efficiently produce reduced order meshes for cell-level thermal analyses using the FEM. We introduced schemes based on weighted grid lines, power density evaluation, and a hybrid mode which scale down mesh sizes while maintaining the characteristics of the thermal profile. The experimental results show a less than 5% error to the temperature of the original, full mesh, with a 90% reduction in the size of the original mesh. This demonstrates the efficiency of the proposed cell-level homogenization algorithms as the size of the mesh is reduced. Statistical analysis shows a reduced order mesh that is 25% of the size of the original mesh captures most of the characteristics of the thermal profile. Consequently, this situations results in a 75% reduction of the simulation time. The proposed smart grid mode is an effective and automated homogenization method requiring no a priori knowledge of the design and combines both the power density evaluation and weighted grid line schemes.

7. REFERENCES

[1] M. M. Waldrop, "The Chips Are Down for Moore's Law," *Nature News*, vol. 530, no. 7589, p. 144, 2016.

[2] C. Xu, S. K. Kolluri, K. Endo, and K. Banerjee, "Analytical Thermal Model for Self-Heating in Advanced FinFET Devices with Implications for Design and Reliability," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 32, no. 7, pp. 1045–1058, 2013.

[3] A. Danowitz, K. Kelley, J. Mao, J. P. Stevenson, and M. Horowitz, "CPU DB: Recording Microprocessor History," *Communications of the ACM*, vol. 55, no. 4, pp. 55–63, 2012.

[4] M. White, "Scaled CMOS Technology Reliability Users Guide," Jet Propulsion Laboratory, National Aeronautics and Space Administration, Tech. Rep., 2010.

[5] J. H. Lau, "Evolution, Challenge, and Outlook of TSV, 3D IC Integration and 3D Silicon Integration," in *International Symposium on Advanced Packaging Materials*, 2011, pp. 462–488.

[6] S. Borkar, "3D Integration for Energy Efficient System Design," in *ACM Design Automation Conference*, 2011, pp. 214–219.

[7] Y. Yang, C. Zhu, Z. Gu, L. Shang, and R. P. Dick, "Adaptive Multi-domain Thermal Modeling and Analysis for Integrated Circuit Synthesis and Design," in *IEEE/ACM International Conference on Computer-Aided Design*, Nov. 2006, pp. 575–582.

[8] P. Li, L. T. Pileggi, M. Asheghi, and R. Chandra, "IC Thermal Simulation and Modeling via Efficient Multigrid-based Approaches," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 25, no. 9, pp. 1763–1776, 2006.

[9] W. Huang, S. Ghosh, S. Velusamy, K. Sankaranarayanan, K. Skadron, and M. R. Stan, "HotSpot: A Compact Thermal Modeling Methodology for Early-Stage VLSI Design," *IEEE Transactions on Very Large Scale Integration Systems*, vol. 14, no. 5, pp. 501–513, 2006.

[10] A. Sridhar, A. Vincenzi, D. Atienza, and T. Brunschwiler, "3D-ICE: A Compact Thermal Model for Early-Stage Design of Liquid-Cooled ICs," *IEEE Transactions on Computers*, vol. 63, pp. 2576–2589, 2014.

[11] T.-Y. Wang and C. C.-P. Chen, "3-D Thermal-ADI: A Linear-Time Chip Level Transient Thermal Simulator," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 21, no. 12, pp. 1434–1445, 2002.

[12] Y.-H. Lee, J. H. Lee, Y. Wang, R. Hsieh, Y. Tsai, and K. Huang, "Consideration of BTI Variability and Product Level Reliability to Expedite Advanced FinFET Process Development," in *IEEE International Electron Devices Meeting*, 2016, pp. 15–2.

[13] C. Baltacı and Y. Leblebici, "Thermal Aware Design and Comparative Analysis of a High Performance 64-bit Adder in FD-SOI and Bulk CMOS Technologies," *Integration, the VLSI Journal*, 2017.

[14] E. Cai, D. Stamoulis, and D. Marculescu, "Exploring Aging Deceleration in FinFET-based multi-core Systems," in *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2016, pp. 1–8.

[15] H. C. Elman, D. J. Silvester, and A. J. Wathen, *Finite Elements and Fast Iterative Solvers: with applications in incompressible fluid dynamics*. Oxford University Press, 2005.

[16] W. L. Briggs, V. E. Henson, and S. F. McCormick, *A Multigrid Tutorial*, 2nd ed. SIAM, Philadelphia, 2000.

[17] D. S. Lo, *Finite Element Mesh Generation*. CRC Press, 2014.

[18] Si2, "LEF/DEF Exchange Format Ver 5.8," 2017, http://www.si2.org/.

[19] C. Geuzaine and J.-F. Remacle, "Gmsh: A 3-D Finite Element Mesh Generator with Built-in Pre-and Post-Processing Facilities," *International Journal for Numerical Methods in Engineering*, vol. 79, no. 11, pp. 1309–1331, 2009.

[20] G. Yip, "Expanding the Synopsys Primetime Solution with Power Analysis," http://www.synopsys.com, 2006.

APPENDIX

A. HEATSINKS

The type "HeatSink" is a specific type used to define and create the surface of a heatsink. The "HeatSink" type is automatically defined to be the top boundary of the circuit domain. This top boundary is associated with the heat transfer coefficient η in (1b). The geometry of the heatsink has two major parts, the fin structure on top and the supporting cuboids on the bottom of the fins.

```
<xCuboid>: x-coordinate of origin of bottom cuboid
<yCuboid>: y-coordinate of origin of bottom cuboid
<zCuboid>: z-coordinate of origin of bottom cuboid
<lengthCuboid>: size of cuboid in x-axis
<widthCuboid>: size of cuboid in y-axis
<heightCuboid>: size of cuboid in z-axis
<lengthFin>: size of each fin in x-axis
<widthFin>: size of each fin in y-axis
<heightFin>: size of each fin in z-axis
<xstartFin>: x-coordinate of center of the first cuboid in x-axis
<ystartFin>: y-coordinate of center of the first cuboid in y-axis
<xnumberFin>: number of fins in x-direction
<ynumberFin>: number of fins in y-direction
<xendFin>: x-coordinate of center of the last cuboid in x-axis
<yendFin>: y-coordinate of center of the last cuboid in y-axis
```

Figure 10: XML Definition of Heatsinks