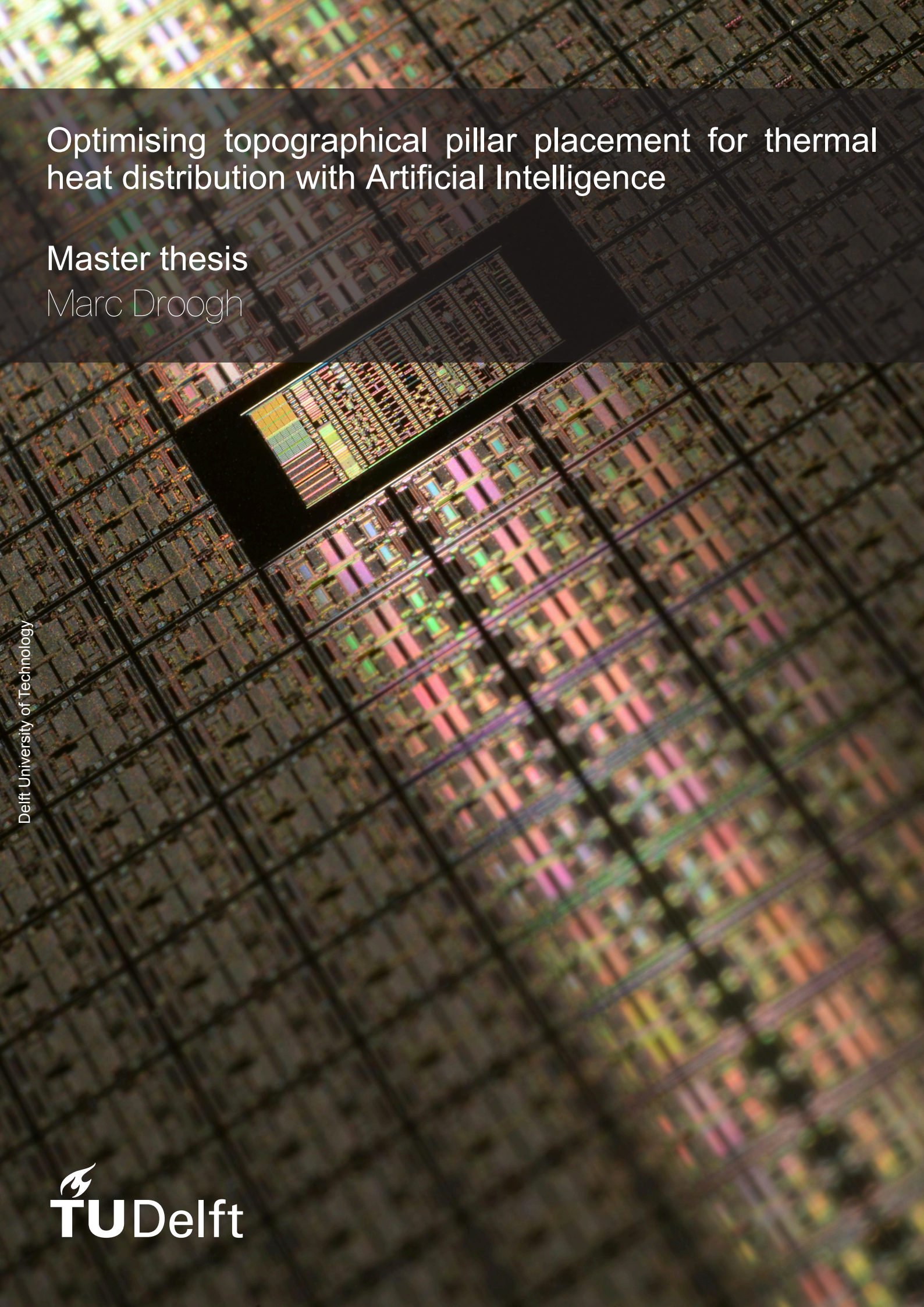
Optimising topographical pillar placement for thermal heat distribution with Artificial Intelligence

Master thesis

Marc Droogh



Delft University of Technology

TU Delft

Optimising topographical pillar placement for thermal heat distribution with Artificial Intelligence

by

Marc Droogh

In partial fulfillment of the requirements for the degree of

Master of Science

In
Computer Science
At the Delft University of Technology
To be defended on Tuesday June 28, 2022 at 10:30 am

Student number:	4699319
Thesis advisor:	Dr. Neil Yorke-Smith (TU Delft)
Daily supervisor:	Sebastijan Dumančić (TU Delft)
External supervisor:	Rene van den Berg (NXP Semiconductors)
External supervisor:	Nick Thomassen (NXP Semiconductors)
Institution:	Delft University of Technology
Place:	Faculty of Electrical Engineering, Mathematics & Computer Science
Research group:	Algorithmics
Project Duration:	November, 2021 - June, 2022

Preface

This is the final product of my master thesis for the Master of Science Computer Science at the faculty of Electrical Engineering, Mathematics and Computer Science, at the University of Technology Delft. This project has been carried out at NXP Semiconductors, a company focused on chips that improve the world.

I would like to thank the whole of NXP Semiconductors and specially my supervisors; Rene van den berg and Nick Thomassen. They have supported me throughout the whole project, taking the time to listen to my ideas and provide constructive feedback. I would also like to thank Denver Desouza from NXP Semiconductors for his unlimited help during the project.

I would like to express my gratitude to my supervisors at the TU Delft, Dr. Neil Yorke-Smith and my daily supervisor Sebastijan Dumančić. Their guidance throughout the project helped me to complete this project successfully.

Finally I would like to thank the people around me. My family, girlfriend and friends for their unconditional support.

Marc Droogh
Delft, June 2022

Contents

Nomenclature

Abbreviations

Abbreviation	Definition
IC	Integrated Circuit
PCB	Printed Circuit Board
BGA	Ball Grid Array
WLP	Wafer Level Packaging
UBM	Under Bump Metallization
TSMC	Taiwan Semiconductor Manufacturing Company
I/O	Input/Output
SHC	Stochastic Hill Climber
SWO	Squeaky Wheel Optimisation
TSP	Traveling Salesman Problem
IPTV	Intellectual Property Test Vehicle
ET	Electrothermal
API	Application Programming Interface

List of Figures

List of Tables

1

Introduction

An Integrated Circuit (IC) is a set of electronic circuits on a piece of semiconductor material, mostly referred to as a ”chip” or die. As electrical equipment became more complex, chips replaced dedicated components in electrical equipment, mainly due to advantages in cost and reliability [1]. Today, an IC is the core of almost every electronic device we use. Gordon Moore already successfully predicted integrated circuits would lead to wonders such as home computers and personal portable communication equipment in 1965 [2]. Moore is also known for his predictions related to the increase of transistors over time, better known as Moore’s law. Moore’s law states that the number of transistors roughly doubles every two years, as can be seen in figure 1.1, this prediction made in 1965 has proven to be accurate. The semiconductor industry and the consumers of its technologies have benefited from the progress made over time. However, multiple problems related to the increase in the number of transistors have occurred. The increasing transistor counts together with the aggressive frequency scaling, result in increased power densities, which can be seen in figure 1.2.

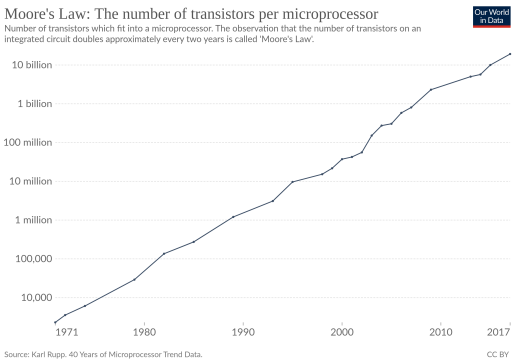


Figure 1.1: Moore’s Law [3]

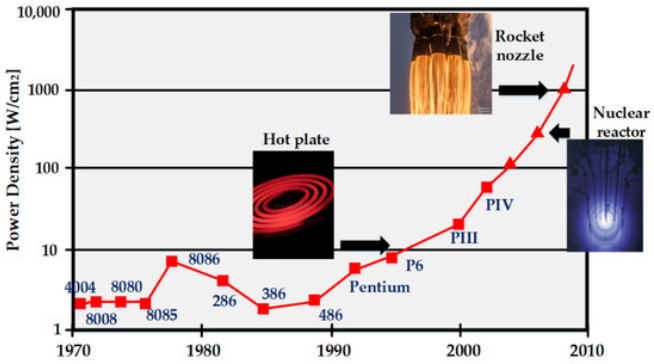


Figure 1.2: Power Density on microprocessors [4]

As a result of the increase in power density on chips, the temperatures on the chips also started to increase. High-performance chips required active cooling, and the repeated heating and cooling of ICs can lead to failures of the package.
When an IC is used in electrical equipment, it is mostly housed inside of a package that protects the chip and supports the electrical contacts which connect the die to the printed circuit board (PCB) of the equipment. These electrical contacts conduct electrical signals between the IC and the PCB on which it is placed. To ensure the reliability and performance of an IC, it is important the temperature of the die stays below a certain threshold. The package of the IC plays an important role in this process [5].
A lot of research has been done on the influence of design strategies and attributes of the package. Some of these studies investigate new materials and measurement techniques for power and temperature. However, it is also proven that the placement of pillars, which connect the die to the substrate of the package in flip-chip design, significantly impacts thermal household on the die [6]. Heat has to be transferred from the die to the outside world, simply put, heat transfer or heat flow. The pillars and their

placement influence the heat transfer. Consequently, a more efficient path of heat conduction results in lower overall temperatures and as a result also in a power decrease.
Currently, due to many other constraints in chip design, the placement of pillars is done by hand and thermal heat distribution is not fully addressed.
This work is focused on optimising the topographical placement of pillars from a thermal perspective by using the power of computing. Multiple algorithmic strategies are carried out and evaluated using high-end simulation software.

1.1. Problem overview

The problem is that the current topographical pillar placements of copper pillars in flip-chip designs are not optimal with respect to thermal heat distribution. In this work, an attempt is made to optimise it using artificial intelligence. This has to be done without requiring significant changes to the other parts of the chip design, while still adhering to hard constraints set by manufacturers.
Throughout the whole design phase, multiple design rules must be adhered to. These are mostly related to performance, reliability and producibility. Customers expect a certain performance and reliability while the production facilities, which are responsible for the fabrication of the designed products, enforce hard constraints on certain design characteristics.
At certain points, alterations can be made to the design without requiring a whole overhaul of the design, but these changes can still significantly impact characteristics like heat flow through the package. The placement of copper pillars in flip-chip designs is an example of such alteration. Altering the topographical placement can reduce the heat on the die significantly. This can be done without requiring too many changes in other parts of the chip.

1.2. Motivation

Designing a chip can be a long and complex process involving many people in different teams working on floor plans and evaluating signal transport routes. Multiple iterations of trial and error are followed by human verification and testing procedures, possibly forcing design engineers to alter the design and repeat multiple design stages. Making the actual IC package is most expensive and exhaustive, therefore computer simulation tools are introduced in every step of the design phase.
In the current design procedure, the peak temperature of the die is important for its reliability and performance, but it is mostly not the main concern. If the peak temperature is below a certain threshold, the design is generally considered satisfactory and the heat distribution is not further optimised. However, in the validation phase, not only the peak temperature is important, but the question is if all parts on the chip are okay. For example, power transistors can usually easily handle temperatures in excess of 200° C, whereas certain digital circuitry might already suffer from excessive leakage issues at 135° C.
Due to the hard design constraints, the impact of the topographical placement of copper pillars on heat distribution and the lengthy task of doing the optimisation by hand, a well-tested algorithm should be able to speed up the process. An algorithmic solution that can help to optimise the heat distribution by altering a specific part of the chip design, while still adhering to hard constraints, can significantly improve the design without requiring many resources or slowing down the design process.

1.3. Research questions

The following questions will be answered in this work:

- How can topographical pillar placement be thermally optimised with the use of Artificial Intelligence?
  - How can hotspots on flip-chip design be reduced by altering the pillar placement?
  - How can a non-uniform heat distribution be improved to move towards a homogeneous case?
  - Under what design conditions is improving heat distribution possible?
- What constraints make it impossible to improve topographical pillar placement?
- How does the size of the pillar influence the pillar optimisation strategy results?

1.4. Contributions

This work provides a solution to the lengthy process of optimising pillar placement. This fully automated solution is able to place, evaluate and correct pillar placements without human interference. The designs outputted by this solution are usable and the thermal characteristics of the solution are clear.
The author would like to inform the user that before this work was started, no such automated flow was in place nor was it clear how to create it. However, a study which investigates different pillar placements and their implications were conducted in 2021 by Denver Desouza [6]. The tools which are used for the pillar placement and evaluation in normal circumstances are centred around human actions and require the expertise of a human to direct the design process. In order to automate the whole process, the author had to explore the limits of the simulation tools involved, combining modern programming heuristics with complex file structures which are interpretable by the simulation tools. This automation can also be used for other projects, which also require simulating chips without human interference.
The different strategies used to optimise the pillar placement were not explored before and were developed by the author in combination with the supervisors. Therefore, the results of these strategies are completely new and require thorough evaluations. The author would like to emphasise that the primary results of the optimisation strategies led to new test setups and a more thorough understanding of the problem.

1.5. Report Layout

Chapter 2 presents a literature study, firstly introducing the relevant background regarding integrated circuit packages, followed by a thorough explanation of heat in these packages. Then, more background on the pillar itself is provided, followed by the characteristics of how a pillar can be used to reduce temperatures on the IC. Finally, an overview of relevant optimisation strategies is given.
Chapter 3 states the practical methods used to tackle the problem. First, the overall approach is discussed. Then, the methods to enforce the strict design rules of section 2.3.3 are explained. In the third section of this chapter, the implementation of the first optimisation strategy, the stochastic hill climber, will be discussed. In the section after that, the implementation of the squeaky wheel optimisation strategy is explained. In the final section of this chapter, an implementation where both strategies are combined is given.
Chapter 4 contains everything related to the experimental design. First, focussing on the objectives of the experiments and then discussing the experimental setup. Finally, the hardware which is used to carry out these experiments is considered.
Chapter 5 displays the results of the conducted experiments. First, focussing on the results obtained on the test setup, distinguishing between two different configurations. Also touching upon different pillar sizes and their influence on the results. Then, the results of the actual product setup are discussed. Finally, the runtimes of the different optimisation strategies are addressed.
Chapter 6 contains the discussion of the thesis. In this chapter, the results are further discussed. Then, the limitations of this work and potential future scope are considered.
Chapter 7 concludes the thesis, answering the research questions.

2

Literature Study

In this chapter, firstly the focus is on the general concepts in chip and package design. These concepts are related to the overall architecture of an integrated circuit package. More specifically, the focus is on two different techniques to do chip-level packaging, namely wire bonding and flip-chip. After that, attention is paid to the overall heat generation and flow in an integrated circuit package, distinguishing between different types of heat transfer. Then, the structure of a copper pillar and its characteristics are discussed. The section that follows describes the heat removal characteristics in the integrated circuit package. Then, the influence of pillar placements, such as geometry and density on heat distribution, is discussed. Finally, different optimisation strategies are discussed, such as local search and squeaky wheel optimisation.

2.1. Integrated Circuit Packages

Integrated circuits can be housed in a package for easy assembly on a printed circuit board (PCB). While an IC can also be placed on a PCB without a package, the package provides chemical and mechanical protection to the IC against environmental conditions and ensures mechanical stability. Without a package, moisture and vibrations can negatively affect the lifetime and reliability of an IC [5]. A package also helps with thermal management and the package provides electrical connections. The die is placed on and attached to, the substrate of the package, this is better known as chip-level packaging [7]. The package itself is attached to the PCB. There are multiple techniques to successfully connect the chip to the package, these techniques are discussed and compared in the following subsections. Connecting the package to the PCB, when considering high-performance applications, is mostly done using the ball grid array (BGA) or variations on BGA techniques [7]. In this document, BGA is considered to be standard.

Depending on the application there are different types of packaging, separable by the manufacturing process and by the size of the package compared to the die. If the packaging is done before the dicing process, the process is called wafer-level packaging (WLP) [8]. In the dicing process, a wafer holding chips are sliced into a big quantity of individual chips [8]. In this work WLP is considered to be standard, the chips considered in this work are produced using this process.

The size of the package is also relevant, the main focus is on chip-scale packages in this document. This refers to packages that are less than 1.2 times the size of the chip or, more strictly defined, have a 1:1 relationship between the chip and package size [9].

2.1.1. Chip Level Packaging

WLP at the chip level is mainly done by using either wire bonding or flip-chip [10]. Key factors in selecting the right technique are cost, form factor and performance [11]. In this subsection, these techniques are described in more detail and then compared to each other.

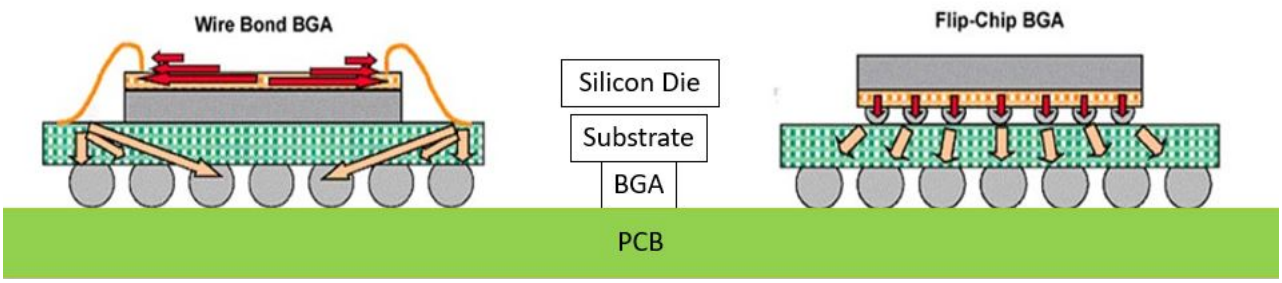


Figure 2.1: Wire Bond and Flip-Chip Package

Wire Bonding
In wire-bond packages, active components of the die are facing upwards, these components are connected to the substrate using wires. These wires are used to transfer input- and output signals between the die and the packaging. Wire bonding can be characterised as a single-point-unit operation, which means that each bond is individually produced. In the fabrication process, a machine identifies the die, transforms and corrects the locations for each bond and moves to each location to produce an interconnection [11].

Flip Chip Design
In flip-chip packages, the die is 'flipped', hence the name flip-chip package. As a result, the active components on top of the die are now facing the substrate. These active components are attached to the substrate by bumps or pillars. These bumps/pillars have the same function as the wires, but can be much smaller compared to their counterpart [12]. Due to the smaller size of the pillars, more pillars can be placed and thus extra connections between the die and substrate can be made. This can have electrical advantages. In the production process, the bumps are formed on the entire wafer before it is diced and the process is never a single-point operation [11]. It is also possible underfill is encapsulated, filling the cavity between the chip and the substrate with a specific material [13]. This underfill corrects thermal expansion mismatches between the die and interconnects. It also provides mechanical stability to bumps/pillars.

Comparison
Wire-bond and flip-chip designs are different due to their characteristics and the difference in production processes. In general, wire bond designs are more flexible, cheaper and easier to produce. These advantages are mostly due to the easier production process of wire bond designs. However, when the number of connections between the die and the substrate increases, flip-chip design can be more cost-effective because the pillars are made in a batch process while wire bonding is a single-point-unit operation. Thus the difference in cost is highly dependent on modelling conditions and substrate technology [11].
The number of connections and size of the substrate are influenced by the difference in technology. While wire-bonding mainly uses the sides of the die to route wires, the flip-chip technology can use the whole surface of the die. As a result, the density of connections between the chip and the substrate is increased for the flip chips. This means the substrate area can also be reduced while the number of connections stays the same [11].
In terms of performance, the flip-chip has many advantages over the wire-bond technology. Due to the shorter interconnects and smaller pins, the inductance can be reduced and thereby increasing the speed of currents flowing through. As a result, for higher frequency applications, the signal integrity is also better in flip-chip designs. Because flip-chip design has no bond wires, there will be less interference of signals to/from the chip and less IR drop due to the short distance from the substrate to chip [11].
Finally, the interconnects in flip chips are more reliable and less susceptible to mechanical failures and fatigue compared to wire-bonds [11].

2.2. Heat flow in IC packages

Microelectronic devices and their packaging are significantly influenced by temperature. Due to trends of minimising die size, while increasing integration and driving up clock speeds, power dissipation throughout the die can further increase die temperatures. Most of this increased temperature is unevenly distributed which leads to hotspots [14]. While for some devices heating is desirable, the generation of heat in an IC package is considered to be 'parasitic', and therefore should be minimised [15]. Joule heating is the heat formed due to current flowing through resistive wiring. In microelectronics, this is considered to be 'parasitic' because it activates damaging mechanisms such as corrosion, fatigue of wires or bonds, thermal breakdown and much more [15]. Since there will always be current on the chip, and therefore there will always be heat generated, the flow of heat plays an important role. The heat generated on the die is transferred through the whole package by conduction. The heat in the package is then transferred to the PCB and ambient mainly in three ways; conduction, convection and radiation [15]. Firstly, the generation and conduction are discussed, and then convection and radiation are considered. Finally, the removal of the heat on the IC is discussed.

2.2.1. Heat generation

Heat is generated in different places of the package, a distinction can be made between three different categories; heat generated by Joule heating in materials, heat generated by analog circuitry, and heat generated by digital circuitry. While there are other effects which increase or reduce the heat generated in the package, such as the Peltier effect and the Thomson effect, this is beyond the scope of this work [16].

Joule heating

Heat in different sections of the package, apart from the IC, is mainly generated by pure Joule heating. Joule heating is the mechanism for heat generation, where heat is generated due to the flow of electrical current through the material [15]. Joule heating is expressed in the formula known as Joule's law:

$$Q = j^2 \rho \tag{2.1}$$

where j is the current density vector in Am^{-2}, ρ the specific electric resistivity in Ωm and Q is the generated heat per unit volume Wm^{-3} [15]. Thus, the power dissipation is a consequence of the so called 'static IR drop', which refers to the reduction in voltage due to the electric resistance of the material it flows through [17].
Furthermore, the thermoresistive effect also plays a role in Joule heat generation. The thermoresistive effect refers to the change in electrical resistivity ρ, which influences the Joule heating, due to a change in temperature [18]. This effect is used to build small and very precise thermometers [18]. The thermoresistive effect can be expressed as a function of temperature T in Celsius:

$$\rho(T) = \rho_0(1 + \alpha T + \beta T^2 + ...) \tag{2.2}$$

where α and β are the temperature coefficients of the material in K^{-1} and K^{-2}, and ρ_0 is referred to as the electric resistivity at a temperature of zero Celsius [15].

Analog circuits

For the heat generated on top of the die, the above representation of heat generation is not accurate. Biasing power is dominant in analog circuits, making static biasing current the main contributor to power dissipation in analog circuits [19]. For analog circuits, and thus analog parts on the die, power dissipation can be captured in the following formula:

$$P = V \times I \tag{2.3}$$

where power P is converted from electrical energy to thermal energy, as a result of supply current I multiplied by the supply level V [19]. Usually, the I is largely determined by the bias current setting and topology of the analog circuitry. The supply level strongly depends on how the analog designer made the circuit, as these structures from the supply line to the ground line can be quite complicated [20].

Digital circuits
For digital circuitry, on the other hand, heat generation is mainly a result of dynamic power dissipation and leakage [21].
Dynamic power dissipation can be seen as the power by all digital nodes switching up and down, and can be captured in the following formula:

$$P = C_L \times V^2 \times f \tag{2.4}$$

where P is the dynamic power dissipation of two components with a load capacitance C_L connected to the inverter output, and f is the frequency at which voltage change V occurs on C_L [21]. There can be small current spikes when a node toggles from 0 to 1 or the other way around. This part of the power is related to the node toggles scales with use.
Power leakage is another dominant factor in power dissipation for digital circuits. Due to the ever-growing integration density, components are getting smaller and this results in leakage becoming a more dominant factor. Sub-threshold leakage is a mechanism in which a single transistor's gate-to-source voltage is below its threshold to start conduction, yet there is current flow from drain to source [21]. For high temperatures on the die, leakage can be as high as half of the total dissipation. As long as circuitry is connected between supply and ground, there will always be leakage.

2.2.2. Heat Conduction

Heat conduction is the most dominant mode of heat transfer on the chip and throughout the package, [22]. Heat conduction refers to the process where a temperature gradient exists within a solid body, the heat energy will flow from a region with a high temperature to another region with a lower temperature [15]. This phenom can be described by Fourier's law as:

$$q = -k\nabla T \tag{2.5}$$

Heat flux vector q in Wm^{-2} for a negative local temperature gradient $-\nabla T$ in Km^{-1} and thermal conductivity k in $W(mK)^{-1}$ [15].
However, when a homogeneous material is considered of 1-D geometry between two endpoints at constant temperatures, the formula can be written as:

$$q = -k \times A \times \frac{\Delta T}{l} \tag{2.6}$$

where q is the heat flux vector in Wm^{-2}, in a material with a thermal conductivity k in $W(mK)^{-1}$ and cross-sectional surface area A in m^2, where ΔT is the difference in temperature between the endpoints in K and the material has a length of l [23].
Considering this formula, keeping the thermal paths short and enlarging the cross-sectional area should help to conduct heat from the die to other parts of the chip. Aiming to use materials with a high thermal conductivity should also help to conduct the heat of the die away and reduce the overall temperature. Also, as most packaging materials are worse thermal conductors for high temperatures, and the power increases with temperature, there is a positive feedback loop. In this loop, the power increases the temperature and the temperature increases the power, resulting in a dramatic increase in temperature [20].
As the transistors in microprocessors shrink toward the nanometer scale, the thermal properties of semiconductor nanostructures become more important for heat conduction [24]. The heat transfer on this nanometer scale can be modelled using Boltzmann's equation [15]. However, this is outside the scope of this work and in this work, only Fourier's law is taken into account considering heat conduction.

2.2.3. Heat Convection

Heat convection is another form of heat transfer, it is the dominant mechanism in terms of heat flows at the package to fluid interface [20]. Convection is defined as the thermal interaction between a surface and an adjacent moving fluid, air for example [25]. Convective heat transfer is more complex than conductive heat transfer as the heat transfer coefficient for convection depends on complex fluid-related factors, such as density differences in the fluid. Convection can be expressed by Newton's law of cooling:

$$q = \alpha A \Delta T \tag{2.7}$$

Where q is defined as the energy flow per second from the surface, with surface area A in m^2, of an object into a fluid in Js^{-1}, α is better known as the heat transfer coefficient in $Wm^{-2}K$ and finally ΔT is defined as the difference in temperature between the solid and the fluid in K.

2.2.4. Heat Radiation

Heat radiation is the final form of heat transfer considered in this work, this mechanism is a dominant factor in terms of heat flow from the package, PCB and encasing if present, to another body and the environment [20]. Thermal radiation is a process where heat is transferred by emission of electromagnetic waves, this is usually expressed as the product of a surface property, the emissivity and the blackbody radiation due to the object's temperature [25]. But, there is also radiation from the surroundings incident onto the object, this leads to the following formula:

$$q = \epsilon \sigma A(T_o^4 - T_s^4) \tag{2.8}$$

where q is defined as the energy flow per second from the object into its surroundings in Js^{-1}, with surface area A in m^2. ϵ is defined as the emissivity of the object, σ is better known as the Stefan-Boltzman constant ($6.67 \cdot 10^{-8} Wm^{-2}K^{-4}$) and T_o, T_s refer to the temperatures of the object and the surroundings respectively in K [25].

2.3. Flip-Chip Interconnects

Flip-chip technology was first introduced by IBM in the early 1960s [26]. Over time, chip-design companies and chip manufacturers have experimented with different interconnecting techniques and materials. First, a popular and influential interconnecting technique called C4 Bumps is discussed. After that, the main interconnecting techniques are considered. In this work specifically, these are the copper pillars. Finally, relevant design rules that are related to the manufacturability of flip-chip interconnects will be considered.

2.3.1. C4 Bumps

In the first generation of flip-chip interconnects, controlled collapse chip connection bumps, better known as C4 Bumps, were used [27]. The name "controlled collapse", comes from the fact that the gap between the chip and substrate is mainly controlled by solder volume [27]. These bumps made of high-lead solder, typically have a pad size equal to 100µm and the target bump height also equals 100µm [28]. The production process consists of optionally redefining the passivisation opening, Under Bump Metallization (UBM) sputtering, spin coating of photoresist, patterning, copper plating and electroplating the solder, stripping of the photoresist, etching and solder re-flow [28].

2.3.2. Copper pillars

Copper pillars are pillar-shaped connections from a semiconductor chip to a substrate, in flip-chip design. Copper pillars were first described as an interconnection technique in a patent from IBM in 2001 [27, 29]. The first pillar described in this patent consists of an elongated pillar, comprising two elongated portions, one including copper and another portion including tin solder [30]. Shortly after that, in 2003 a patent was issued to Fransisca Tung, in this patent she specified a similar design with a copper column height of 55µm with a total bump height of 80 to 120 µm [27].
Intel added a diffusion layer and an external wetting layer to the metal post in 2007, placing the solder on top of this structure [27, 31]. The first step for first-level flip-chip interconnect using copper post and eutectic solder was made in 2006, when Intel showed the integration of copper posts on the microprocessor die side [27, 32].

The design of Taiwan Semiconductor Manufacturing Company (TSMC), is similar to that of IBM's and is considered as the main design in this work. It uses an elongated Cu pillar structure as base and for the pillar height to solder height, a ratio of 2:1 is used [34].
The use of copper pillars is preferable to C4 bumps, possibly with solder caps, because it can result in a higher pin count and smaller spacing between pads. This can then potentially shorten the adjacent solder C4 bumps [28]. Compared to C4, copper pillars also suffer less from problems related to solder, such as reduced signal impedance and solder bridging[27].
Apart from connecting the die and the substrate physically and routing I/O signals from the die to other

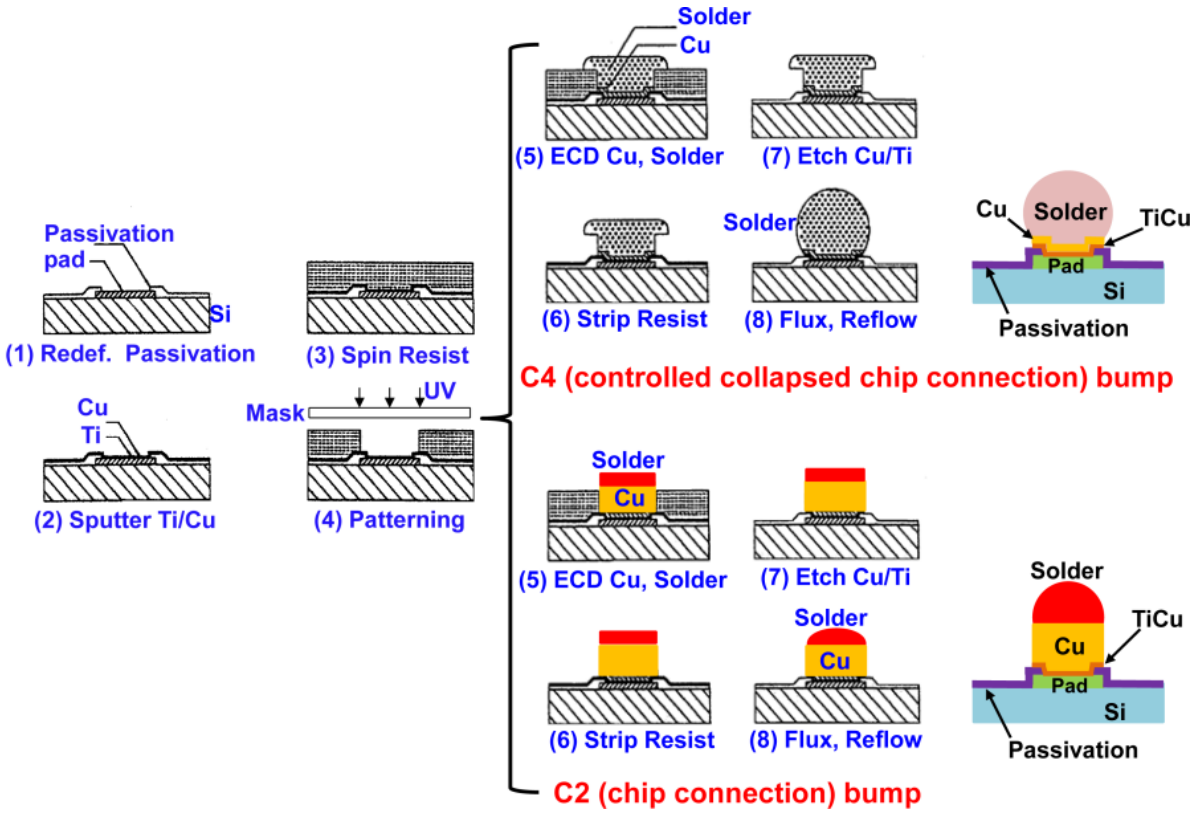


Figure 2.2: C4 bump and C2 bump production process [28]

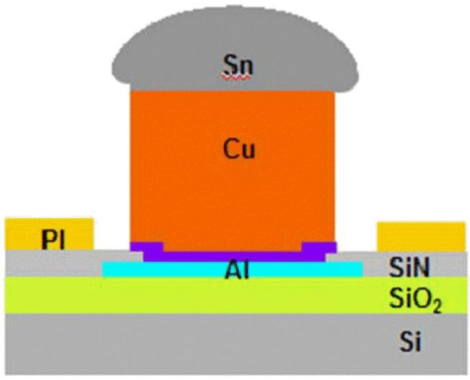


Figure 2.3: IBM's MPS-C2 [29, 33]

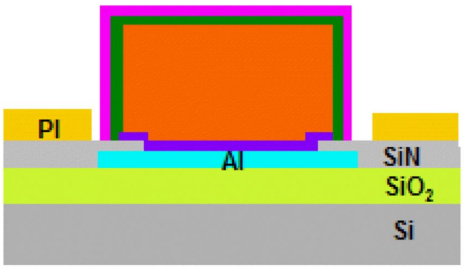


Figure 2.4: Intel's Cu post patent [31, 33]

layers, interconnects such as copper pillars are great heat conductors, reducing the temperature on the die [6]. As observed in 2.2.2, the thermal conductivity of material plays an important role in heat conduction, the thermal conductivity of copper is roughly 7 times higher than solder's thermal conductivity (400 W/m K and 55-60 W/m K respectively) [28]. Therefore, copper pillars are expected to conduct more heat at the same temperature difference, or conduct the same amount of heat against a lower delta temperature, compared to C4 bumps.

The first steps in the production process of copper pillars are the same as for the C4 bumps, but the electroplating of solder is replaced by the electroplating of copper [28]. Electroplating of copper is followed by electroplating the solder cap and reflowing the solder with flux [28]. The differences in the production process can be observed in figure 2.2. C2 bumps are not controlled collapsed and therefore named C2 instead of C4.

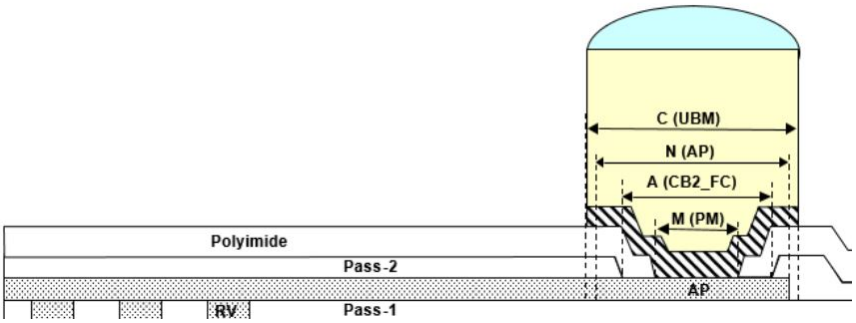


Figure 2.5: TSMC elongated pillar design [34]

2.3.3. Manufacture Design rules

In chip design, many design rules are not necessarily enforced due to the functioning of the chip itself, but rather because of manufacturability. For example, currently the manufacturer TSMC does not allow flip-chip designs to have different pillar diameters on the same silicon die [34]. These design rules are important to be taken into consideration, while some of them are too complex to address in this work, others are relatively easy to model. As materials, bump configurations and many more of these characteristics are kept constant throughout the project, these design rules do not influence the experiments. Design rules that are related to the topographical placement of copper pillars are only considered. Rules related to the function of certain pillars, such as input and output, are ignored.
Using the rules set by the manufacturer and the expertise of NXP Semiconductors, the following list contains the rules to which topographical pillar placement has to adhere in this work [34]:

- Minimum distance between pillars: Each of the pillars should have a minimum distance to all other pillars. This distance scales with the pillar diameter
- Sensitive areas: Certain areas of the chip should be pillar-free
- Side of the chip: Pillars have a minimum distance to the side of the chip
- All pillars are the same: All pillars should have the same diameter, height and stack-up

2.4. Heat Removal Characteristics

In principle, the process of removing heat from the die, utilises the effects specified in sections 2.2.2, 2.2.3 and 2.2.4. The heat, which is mainly generated on the die, should be transferred to the substrate as effectively as possible. From there the heat is transferred through the BGA into the PCB and further into the encasing if present. Then, the heat is dissipated into ambient by convection and radiation. As formulas 2.7 and 2.8 show, the magnitude of these transfer mechanisms scale with area. Thus cooling is mainly about increasing the cooling area, this is also the reason why heat sinks have fins for instance. Since the pillars are connected to the die, the most prominent way of heat transfer is through conduction. Because the top of the pillar is connected to the die, and the bottom is connected to the substrate, it is assumed the heat is conducted in the Z-direction and formula 2.6 can be used.
The power dissipation q is given, and scales with temperature. But there is no direct control over it since it is a result of the operating chip. The thermal conductivity k is determined by the material properties and is temperature dependent. However, the usage of materials is given and only dependent on the temperature. When l is taken as the height of the pillar, thus determined by the chosen interconnecting technology, l is also constant. This means that in the 1D approach, only the cross-sectional area A of the pillars can be increased to reduce the delta T. This can be done by placing more pillars with equal size or by placing the same amount of pillars with a higher diameter. A combination of the two would also work.
However, when the problem is approached in 3D and the distance l is not necessarily the height of the pillar, but rather the shortest path from areas on the die with a high power density to the lower layers, moving around pillars influences this l. Thus, by moving a pillar to a location where there is a high power density and a resulting high temperature, the l is decreased and the delta T decreases accordingly. By placing pillars at the right positions, we can avoid lateral paths by offering direct paths

to conduct the heat and as such lower delta T.
Finally, the resistance of the whole layer in which the pillars are placed should be taken into account since this influences the layer's thermal resistance. It is for example possible the underfill material between the pillars also conducts heat from the chip to the lower layers. The total thermal resistance of this layer can be modelled with a method described by Yang Liu et al. [35] as:

$$\frac{1}{R_{layer}} = \frac{1}{R_{Cu}} + \frac{1}{R_u} \tag{2.9}$$

where R_{layer} is the thermal resistance of the whole layer, R_{Cu} is the thermal resistance of Cu pillars and R_u is the thermal resistance of the underfill [35]. R_{Cu} and R_u can be expressed as:

$$R_{Cu} = \frac{l}{k_{Cu}AX} \tag{2.10}$$

$$R_u = \frac{l}{k_u A(1-X)} \tag{2.11}$$

here A is the area, l is the thickness and X is the area ratio of Cu pillar to underfill and k is the thermal conductivity of the materials [35]. Thus the total parallel resistance can be formulated as:

$$\frac{1}{R_{layer}} = \frac{k_{Cu}AX}{l} + \frac{k_u A(1-X)}{l} \tag{2.12}$$

From this, it can be concluded that if the pillars to underfill ratio is kept constant, thus the amount of pillars stays the same and the total surface area is constant, the thermal resistance of the whole pillar layer with underfill is also constant. The thickness of the layer and the materials used are also considered to remain constant. Therefore, any changes in temperature in experiments, when the amount of pillars used is kept constant, cannot be the result of a change in the thermal resistance of the pillar layer.

2.4.1. Pillar geometry

As explained in section 2.3.2, the geometry of the pillars can vary and this also influences the heat removal capability of a pillar. When the formula 2.6 is taken into account again and considering only the pillar itself, increasing the width of the pillar should increase the cross-sectional area. Therefore, the amount of heat which can be conducted by a single pillar will increase when its diameter is increased. The height of the single pillar also influences the heat flux vector q in formula 2.6, the height is specified in this formula as l. As the height of the pillar is increased, the heat flux vector q decreases at a constant delta T. However, in experiments, the height is kept constant. Finally, the material of the pillar is considered, as specified in section 2.3.2, the pillar is an elongated copper pillar with a solder cap. Pillar height and material stack-up will influence the results. For example, if the height of the pillar is dramatically increased, it will surely be the dominant contributor to the thermal changes. However, within a certain technology node, these two parameters, pillar height and material stack-up, are a given fact and as such not part of the study.

2.4.2. Pillar density

Pillar density refers to the number of pillars placed within a certain area. Increasing the density in a certain area would mean increasing X in equation 2.12. Assuming that the thermal conductivity of copper is higher than the underfill material, increasing X should reduce the parallel resistance of this area, and therefore increase the heat it can conduct. Therefore, it is expected areas with a higher pillar density should conduct more heat from the chip to the lower layers. This is confirmed by the experiments of Yang Liu et al. [35] and Denver Desouza [6].

2.4.3. Pillar placement

The only useful research on the influence of the pillar placement on heat removal has been done in 2021 by Denver Desouza [6]. As explained at the beginning of this section, influencing the amount of heat which can be conducted and removed, can be done by placing a pillar at a different location. An interesting observation was made in the research conducted by Denver Desouza [6], while the density of pillars in a certain area was higher, namely placing four pillars compared to one, the one pillar was able to keep the temperature on the chip lower because it was placed at the centre of the hotspot.

2.5. Optimisation techniques

Due to the complexity of flip-chip designs and the complex process of heat generation thereof, general optimisation problems will be used to address this problem. These optimisation strategies can be used without making assumptions about the underlying problem. Thus, the optimisation strategies should not take circuit design or power usage into account, but rather focus on the resulting temperature on the chip. In the next subsections Hill Climbing and Squeaky Wheel Optimisation are described. These techniques are chosen because they have been shown to yield good empirical results in real-world complex problems [36–38]

2.5.1. Hill Climbing

Hill climbing is a local search optimisation technique. Local search is an approach that has been met with empirical success in difficult combinatorial optimisation problems [36]. Hill climbing approaches start with an initial solution and find better solutions by making small changes to the initial solution. These perturbations are said to be in the neighbourhood of the initial solution [39]. The neighbourhood of a solution is the set of all solutions which can be made by slightly altering the current solution.

Simple Hill Climber

A simple hill climber starts with an initial random solution. Giving a solution, the simple hill climber considers all perturbations in the neighbourhood of the current solution. These perturbations are obtained by making small changes to the previous solution according to a predetermined set of rules. Simple hill climbers evaluate all possible perturbations and continue using the best among them. Iteratively, the Hill Climber can converge towards to optimal solution. If there is a local optimum, however, the Hill Climber can get stuck here without finding the optimal solution.

Consider for example the two environments below. A Hill Climber is tasked to maximise the Y-coordinate in an environment. It can do so by taking a step towards a lower X-coordinate or a higher X-coordinate. In figure 2.6, the Hill Climber can easily find the maximum Y-coordinate and end up at the red dot. If the Hill Climber for example starts at the X-coordinate of 1.0, it evaluates going towards the lower X-coordinate of 0.0 and thus obtaining a Y-coordinate of 1.0 or going to the higher X-coordinate of 2.0 and thus obtaining a Y-coordinate of 0.018. Once the Hill Climber reached the X-coordinate of 0.0 and obtains the maximal value 1.0 for the Y-coordinate, going to the lower X-coordinate of -1.0 or the higher X-coordinate of 1.0 is both worse than its current position. When all options are worse than its current location, the Hill Climber halts.

Now consider the environment of 2.7. This environment has two local optimums shown by the red dots on the graph. While point $(3.0, 2.0)$ is the global optimum, a Hill Climber can get stuck at the local optimum of point $(0.0, 1.0)$. Let the Hill Climber for example start at the X-coordinate of 1.0. The Hill Climber will evaluate the two options of moving towards a lower value of the X-coordinate or the higher value of the X-coordinate. When it considers the lower value of the X-coordinate of 0.0 and obtains a Y-coordinate of 1.0, it will prefer this location over the higher value X-coordinate of 2.0 where a Y-coordinate of 0.754 is obtained. As a result, the Hill Climber moves towards the red dot at location $(0, 1.0)$ and stays there. Going towards the X-coordinate of -1.0 or 1.0 is worse than staying at its current location. Therefore the Hill Climber is stuck in the local optimum of $(0, 1.0)$ and will never reach the global optimum at point $(2, 3.0)$.

Stochastic Hill Climber

The simple Hill Climber described above considers all possible solutions in the neighbourhood. When not all perturbations are considered but only one perturbation is considered at random, the hill climber is said to be stochastic [40]. To reach optimality, the stochastic hill climber compares the current solution to a random perturbation in the neighbourhood of the current solution. When it for example starts at the X-coordinate of 1.0 again in the environment shown in figure 2.7, it does not consider the lower value of the X-coordinate and the higher value of the X-coordinate at the same time, but only one of them randomly. It could for example be the case that it randomly chooses to evaluate the higher value of 2.0 as X-coordinate. As a result, the Y-coordinate corresponding to this value is higher than the Stochastic Hill Climber's starting position and the Hill Climber will move to that location. The Stochastic Hill Climber will consider randomly the X-coordinate values of 1.0 and 3.0. But since the Y-coordinate value corresponding to the X-coordinate value of 1.0 is lower than its current location at X-coordinate 2.0, it will never move back. Therefore, it will end up in the global optimum of $(3.0, 2.0)$. But if the Stochastic

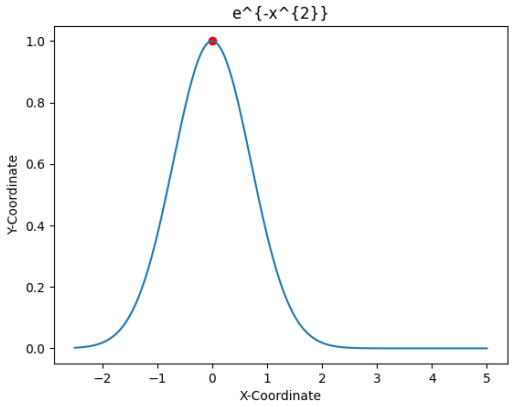


Figure 2.6: Simple Hill Climber Environment

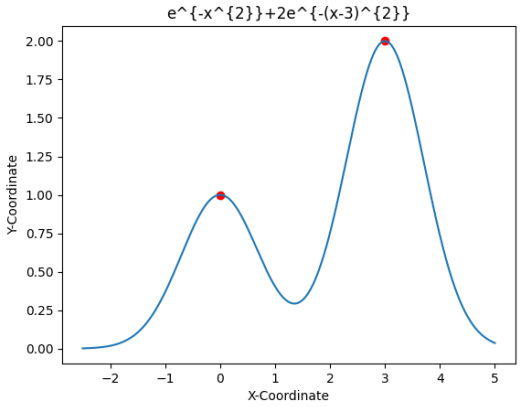


Figure 2.7: Hard Hill Climber Environment

Hill Climber choose to evaluate the lower value of the X-coordinate in the above example, it would still have ended up in the local optimum at point $(0.0, 1.0)$.
There are multiple strategies which can be used to stop the Stochastic Hill Climber from converging to a local optimum. One of them is to randomly restart, placing the Hill Climber at a different starting point. While doing this, it is important to keep track of the best solution found so far. Another strategy which can be used is to help the Stochastic Hill Climber from converging to a local optimum by accepting a new solution by chance, even if it is not better than its current location. This way the Stochastic Hill Climber explores more of the search space [41]. For example, if the Stochastic Hill Climber converged to the local optima point of $(0.0, 1.0)$ in the environment shown in figure 2.7, it can still reach the global optimum. If the Stochastic Hill Climber for example considers going to the X-coordinate of 1.0, while this is not better in terms of Y-coordinate compared to its current location, it takes it anyway by chance. As a result, the Stochastic Hill Climber can now consider the X-coordinate of 2.0 and converge to the global optimum of point $(3.0, 2.0)$.
Because the Stochastic Hill Climber does not consider all the possible perturbations in the neighbourhood, the stopping criteria are less well defined. The Stochastic Hill Climber could keep track of all possible solutions and have similar stopping criteria as the Simple Hill Climber, but for some cases evaluating all perturbations can be extensive. Another example is to let the Stochastic Hill Climber continue until it has evaluated a certain number of perturbations without finding a more optimal one. The downside of this is that there could be a better solution in the neighbourhood but the Stochastic Hill Climber never evaluated it by chance and it stops too early.

2.5.2. Squeaky Wheel Optimisation

In Squeaky Wheel Optimisation (SWO), a greedy optimisation strategy is used to construct an initial solution. This solution is then analysed to identify trouble spots. Based on these trouble spots, elements of the solution, which are likely to improve the overall objective score, are altered [42]. SWO is a general technique for optimisation and has been applied to a wide range of problems in multiple domains, such as scheduling problems and graph colouring problems [37, 38]. SWO typically works well for real-world problems and is relatively easy to understand compared to other techniques. The optimisation process is a continuous cycle of three phases shown in figure 2.8, namely: Construct, Analyse and Prioritise. The cycle continues until some limit or until an acceptable solution is found.
In the construction phase, a solution is constructed by a greedy algorithm. The solution obtained is expected to be optimised for a particular element of the problem. So not necessarily for the overall objective function, and therefore early solutions are expected to contain a lot of sub-optimal decisions and can even violate hard constraints [42].
In the analysis phase, the solution constructed is analysed, identifying poor and well-performing elements. This can be done by assigning a numeric ”blame” to elements that contribute to flaws in the current solution. For example, late tasks in a scheduling problem where lateness is minimised would receive a high value of blame. The analysis phase is a key principle of the SWO, it can also reveal information about the local problem structure by identifying poorly performing elements, and well-working

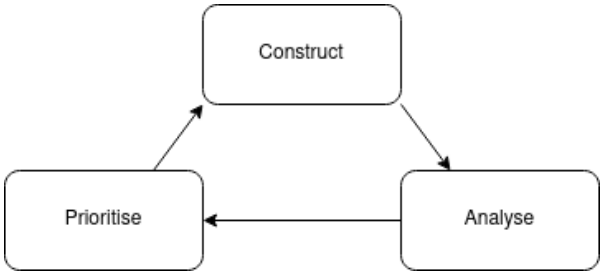


Figure 2.8: Squeaky Wheel Optimisation cycle

elements [42].
In the prioritisation phase, the blame factors assigned in the analysis phase are used to modify the previous sequence of problem elements to deal with. The elements that received the most blame are moved towards the front of the sequence. By moving them towards the front of the sequence, these elements will be handled sooner in the next construction phase [42].
A popular problem that is used to explain the SWO in more detail is the Traveling Salesman Problem (TSP). As the name suggests, the TSP is inspired by a travelling salesman who wants to visit all of the towns once and finish in the town where he started [43]. But, the problem is to find the path to visit all the towns with the shortest possible path. This is a problem which is easy to recognise today, for example when a delivery van starts at a warehouse and has to deliver packages at certain locations, it ends up back at the warehouse. It is proven that this is an NP-Hard problem [43]. Therefore, finding the optimal solution in polynomial time is impossible and verifying that a solution is optimal is also not possible in polynomial time. Consider figure 2.9 as an example of a TSP. Every node is a town, and for every town, there is a path to every other town. In the next subsections, the Squeaky Wheel Algorithm is applied to the example.

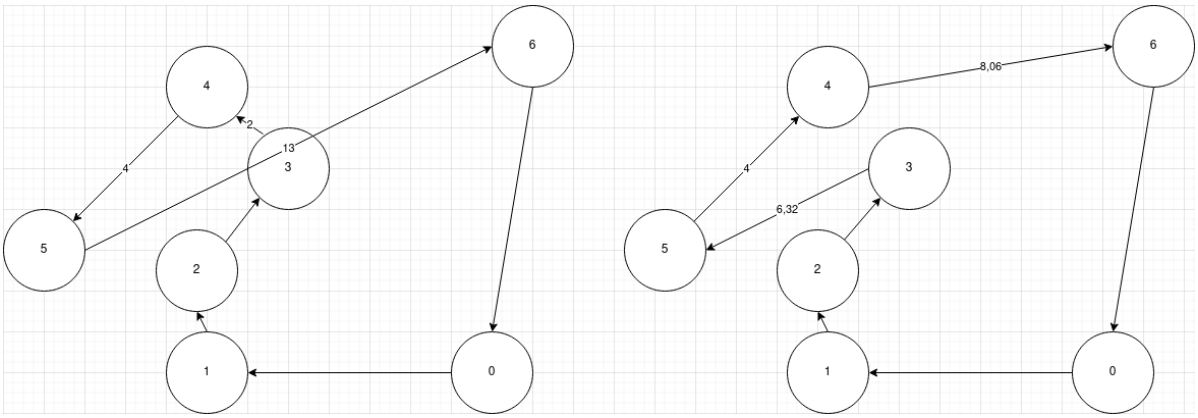


Figure 2.9: Traveling Salesman Problem Example

Construct
A greedy solution to this problem is to start at a random town and travel to the closest town that has not been visited yet. Continue until there are no more towns and finally travel to the starting town. Consider the TSP in figure 2.9, randomly start at town 0. From this location, town 1 is the closest not visited town, this town is visited next. This process continues until town 6 is visited. Now there are no more towns which have not been visited and the path ends by visiting town 0 again. Resulting in the ordering 0-1-2-3-4-5-6-0, this can be observed on the left-and side of the figure.

Analyse
In this phase, the solution generated by the greedy algorithm is analysed. The length of the different paths can be used as a blame factor. In the example, the path from town 5 to town 6 is the longest. That's the reason this path gets the most blame.

Prioritise
In this phase, the blame assigned to each of the paths can be used to obtain an order of paths which should be altered. This can be obtained by ordering all of the paths on blame and thus length. Starting with the longest path 5-6, while having the shortest path 1-2 at the end.

Construct
Now in the construction phase, the order obtained in the prioritisation phase can be used to construct a new solution. Since the longest path was between towns 5 and 6, one of these towns should be placed in a different location of the order in which the towns are visited. The SWO could for example do this by swapping two towns which are after each other in the ordering. For example, if the towns 4 and 5 are swapped, the final ordering becomes 0-1-2-3-5-4-6-0. This can be observed on the right-hand side of figure 2.9. When we compare the lengths of the paths in the figure, the starts of both paths are the same. But travelling from town 3 to town 6 in the original route took 19 units, while in the new order it takes 16.38 travel units. Thus, the SWO reduced the length of the path TSP.

3

Methodology

In the current designs of chip packages, the topographical placement of the copper pillars is not optimised for thermal heat distribution. This can result in high temperatures on the die. These high temperatures can reduce the performance and lifetime of a chip. In this chapter different approaches to optimise pillar placement will be discussed, these approaches are mainly related to the effect of pillar placement discussed in section 2.4.3. First, the general approach is used which is based on the theory from section 2.4. Then, the incorporation of the design rules of section 2.3.3 is considered. After that, the approach will be combined with the optimisation strategies addressed in section 2.5, to come up with potential pillar maps which adhere to the design rules. These strategies use the stochastic hill climber characteristics, then the squeaky wheel optimisation techniques and finally a combination of the two.

3.1. Approach

The main goal is to optimise the distribution of the heat using the placement of the copper pillars. Optimising the heat can be interpreted in multiple ways, one way is lowering the maximum temperature on the die. However, while reducing the maximum temperature is important for the reliability of the chip, the heat should also be distributed evenly over the chip. In other words, the differences in temperatures between hotspots and colder spots on the chip should be reduced and the heat should flow in a uniform matter.

The problem is approached by first focusing on the hottest points on the chip. The most effective way of conducting the heat away from these hotspots is by placing a pillar right at the centre of the hotspot. Consider for example the heatmap shown in figure 3.1, there are two hotspots, one in the top right and one in the middle of the chip. The optimal location of a pillar would be in the middle of the hotspot with the higher temperature in the top-right, shown in figure 3.3. After placing this one pillar, the heatmap changes to the heatmap shown in figure 3.3. The maximum temperature was reduced by 10.77 degrees Celsius.

However, this process can be complex when there are multiple hotspots, because of the mutual heating, resulting in other hot areas. Consider for example the region with the pink outline in figure 3.3. This section has an increased temperature compared to the other regions outside the hotspots. The increased temperature is a result of mutual heating in this section. This can then influence the optimal location to place a pillar. Also, by adding as many pillars as possible below a hotspot, and thus increasing the pillar density in this section, more of the heat can be conducted away from the substrate as explained in section 2.4. The second point of attention is about the sections on the chip which are not necessarily hotspots created by extensive power usage of the chip, but places where there is a very low density of pillars. These sections do not get as hot as the hotspots, but tend to heat up when ignored. Therefore, placing a single pillar at the hottest point, mostly the centre, of such a section should stop it from getting increasingly hotter. Also, keeping these sections from heating up should result in a more uniform distribution of the heat.

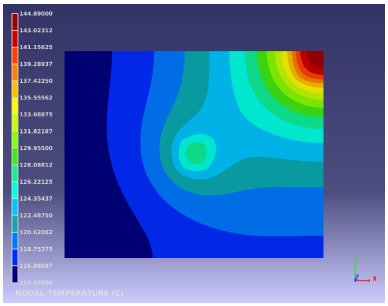


Figure 3.1: Heatmap - no pillars dual hotspot heatmap

Figure 3.2: Pillarmap - one pillar dual hotspot heatmap

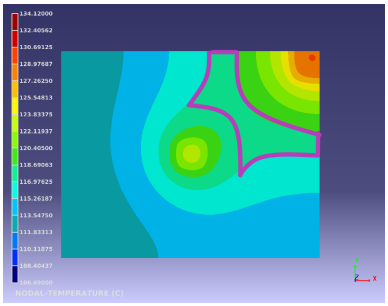


Figure 3.3: Heatmap - one pillar dual hotspot heatmap

Finally, by simulating the changes made to the pillarmap and analysing the results of the heatmap, hotter sections can be reduced in temperature and the effects of adjusting the pillarmap measured. This way, iteratively the solution can be optimised and bad placement of pillars corrected.

3.2. Enforcing design rules

The design rules specified in section 2.3.3 should be adhered to because then results are valid. To make sure feasible pillar maps are created, the design rules are encoded in the algorithm. A coordinate system is used to keep track of the pillars and the zones where no placement is allowed. The preciseness of the system can be increased but the runtime of adjusting the feasibility regions will also increase. In this system, every pillar in the list of pillars has an x- and y-coordinate. This makes it possible to write solutions to the right pillar output format which is taken in by the simulation tool.
The algorithm to check if a pillar can be placed at a certain location starts by creating a feasibility map of the whole chip. First, the size of the two-dimensional array to hold all the feasibility values is calculated. For the x-dimension, this is the maximum x-value of the region where the pillars are placed minus the minimum x-value. This value is multiplied by 10 to the power of the precision. If the precision is, for example, 3 then for each millimetre there are 1000 values thus the precision is up to the micron level. Then for the y-dimension, the size of the array is calculated in the same way but now using the maximum and minimum y-values of the region where the pillars are placed. The size of the array in the x-dimension and y-dimension is used to create a two-dimensional array that will be called the feasibility map. Then for each of the regions specified in the forbidden regions, all the values in the feasibility map which correspond to this region are set to 1. This indicates that at these locations no pillars should be placed. Then for all the pillars which are specified in starting pillars, the region around this location is set unfeasible by the set unfeasible pillar function. Finally, the feasibility map is returned. In pseudocode this results in the following algorithm:

Algorithm 1: Create feasibility map

procedure create_feasibility_map(dimensions, forbidden_regions, starting_pillars, min_distance, precision)
    $x\_size \leftarrow (dimensions.max\_x - dimensions.min\_x) \times 10^{precision}$
    $y\_size \leftarrow (dimensions.max\_y - dimensions.min\_y) \times 10^{precision}$
    $feasibility\_map \leftarrow$ Array(x_size)(y_size)
    for $region$ *in* $forbidden\_regions$ **do**
        $feasibility\_map[region.min\_x : region.max\_x, region.min\_y : region.max\_y] = 1$
    for $pillar$ *in* $starting\_pillars$ **do**
        $set\_unfeasible\_pillar(pillar, feasibility\_map, min\_distance, 1)$
    return ($feasibility\_map$)
end procedure

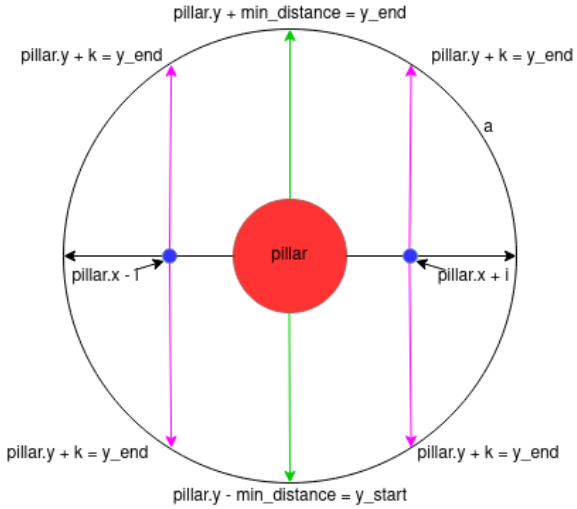


Figure 3.4: Feasibility map visual representation

The set unfeasible pillar function takes in a parameter called min distance. This is the minimum distance between the centre of pillars. The function should increase all values in the feasibility map which are closer than the min distance to the pillar. This is done by first calculating the minimum and maximum y-values of the locations which need to be set to unfeasible and have the same x-value as the input pillar. This is done by subtracting and adding the min distance to the pillar's y-value. While this is done the values are corrected if they get smaller than zero or larger than the maximum y-value of the feasibility map. All points between and including these points are then increased by the delta. In figure 3.4 this is represented by the green line in the middle. Then the algorithm loops from 1 until the min distance. In this loop, the value k is calculated by taking the square root of the min distance squared minus the iteration value squared. If the x-value of the pillar added to the iteration value is smaller than or equal to the maximum x-value of the feasibility map; the x-value of the points that need to be increased by the delta is calculated. This is done by taking the minimum of the pillar value added to the iteration value or the maximum x-value of the feasibility map. In figure 3.4 this is represented by the blue point on the right-hand side of the pillar. Then the minimum and maximum y-values which need to be increased by the delta for this x-value are calculated. In figure 3.4 this is represented by the pink arrows at the right-hand side of the pillar. All the values on this line are increased by the delta. Now, the same is done for the left side of the pillar. Instead of adding the iteration value to the x-value of the pillar, it is now subtracted. This is represented by the blue dot at the left-hand side of the pillar in figure 3.4. All points on the pink arrow at the left-hand side of the chip are increased by the delta. When the loop is finished, all the values in the big circle around the pillar are increased by the delta. Finally, the adjusted feasibility map is returned. This results in the following pseudocode:

Algorithm 2: Set unfeasible Pillar

```
procedure set_unfeasible_pillar(pillar, feasibility_map, min_distance, delta)
    y_start ← max(pillar.y − min_distance, 0)
    y_end ← min(pillar.y + min_distance, length(feasibility_map))
    feasibility_map[pillar.x, y_start : y_end] += delta

    for i in range(1, min_distance) do
        k ← sqrt(min_distance^2 − i^2)
        if pillar.x + i <= length(feasibility_map) then
            x_coordinate ← min(pillar.x + i, length(feasibility_map)
            y_start ← max(pillar.y − k, 0)
            y_end ← min(pillar.y + k, length(feasibility_map[x_coordinate])
            feasibility_map[x_coordinate, y_start : y_end] += delta
        if pillar.x − i >= length(feasibility_map) then
            x_coordinate ← pillar.x − i
            y_start ← max(pillar.y − k, 0)
            y_end ← min(pillar.y + k, length(feasibility_map[x_coordinate])
            feasibility_map[x_coordinate, y_start : y_end] += delta
    return (feasibility_map)
end procedure
```

When the location of a pillar is changed, the new location is checked for feasibility. This is done by first checking if the x-value of the pillar is bigger than 0 and does not exceed the maximal x-value. The same is done for the y-value of the pillar. Then the location of the pillar is used to check if the value of the feasibility map is not bigger than the delta. If this is all true, the location is considered to be feasible and the function returns true. The following is the pseudocode of the algorithm:

Algorithm 3: Is valid location

```
procedure is_valid_location(pillar_x, pillar_y, dimensions, feasibility_map, delta)
    if pillar_x > dimensions.max_x or pillar_x < 0 then
        return False
    if pillar_y > dimensions.max_y or pillar_y < 0 then
        return False
    if feasibility_map[pillar_x, pillar_y] > delta then
        return False
    return True
end procedure
```

3.3. Stochastic Hill Climber

As explained in section 2.5.1, a hill climber starts with an initial solution, in the context of this work this means a random placement of pillars. The hill climber then improves by making small changes to the existing solutions, thus making small changes to the pillarmap. The hill climber is only allowed to move pillars which are labelled as 'dummy' pillars. Since it is stochastic, the hill climber moves the pillars randomly and then checks if the new position is better, and accepts the new location by chance. To determine if a new position is better, the hill climber uses the heatmap generated by simulating the setup without any pillars. When the new location has a higher temperature on the earlier obtained temperature map, the new location is considered to be better and the pillar is moved accordingly. To prevent the solution to be stuck in a local optimum, the new viable locations are accepted by chance, this means that even if a new location is theoretically worse, there is a small chance the pillar is moved anyways. There is a predetermined stopping criterion in place, after a certain number of tries without improvement, the hill climber considers the result to be optimal. The strategy does not re-simulate and mainly optimises the maximum temperature on the whole chip.
In practise the algorithms loops for the predetermined number of optimisation steps. In this loop, the step size is updated so it decreases over time. Then, for all pillars it is first checked if it is a dummy

pillar. If this is the case, a random value between minus step size and the step size is generated to move in the x-direction. The same is done to move in the y-direction. The new location is calculated by adding the moves to the current location. The delta is initially set to 0. If the distance to the new location is smaller or equal to the min distance the delta is set to 1. Then it is checked whether the new location is valid. If this is the case, a random value is generated between 0 and 1, this value will later be used to move it by chance. The temperature under the new location is obtained. If the new temperature is higher then the temperature of the current pillar or the randomly generated value is smaller then epsilon, the pillar is moved. To move the pillar, the feasibility map is altered by setting the surroundings of this pillar to unfeasible but with a delta of -1. Then the x-,y- and temperature-values of the pillar is updated. Then the function set unfeasible pillar is used again but now with a delta of 1. Finally, the pillars and feasibiliy map is returned. This results in the following pseudocode:

Algorithm 4: Stochastic Hill Climber

```
procedure stochastic_hill_climber(pillars, dimensions, feasibility_map, optimisation_steps,
input_step_size, min_distance)
    for i in range optimisation_steps do
        step_size ← input_step_size − (input_step_size × i/optimisation_steps)
        for pillar in pillars do
            if pillar.kind == dummy then
                move_x ← rand(−step_size, step_size)
                move_y ← rand(−step_size, step_size)
                new_x ← pillar.x + move_x
                new_y ← pillar.y + move_y
                delta ← 0
                distance ← √(move_x² + move_y²)
                if distance <= min_distance then
                    delta ← 1
                if is_valid_location(new_x, new_y, dimensions, feasibility_map, delta) then
                    eps ← rand(0, 1)
                    new_t ← calculate_temperature(new_x, new_y)
                    if new_t > pillar.t or eps < epsilon then
                        feasibility_map ←
                        set_unfeasible_pillar(pillar, feasibility_map, min_distance, −1)
                        pillar.x ← new_x
                        pillar.y ← new_y
                        pillar.t ← new_t
                        feasibility_map ←
                        set_unfeasible_pillar(pillar, feasibility_map, min_distance, 1)
    return (pillars, feasibility_map)
end procedure
```

3.4. Squeaky Wheel Optimisation

As explained in 2.5.2, squeaky wheel optimisation starts with an initial solution and then improves it by addressing trouble spots. In the context of pillar placement, trouble spots are hotspots. As a result, the setup has to be re-simulated when changes to the pillarmap are made. As a result, this strategy should also distribute the heat better in sections where there are no hotspots. Simulating the setup is visually represented in the figure 3.5.

In the first phase of this strategy, the construction phase, an initial solution has to be created. Because this strategy is specifically aimed at placing the pillars at the centre of the hotspot, the initial solution is an empty map. If the initial solution was non-empty, it could be the case that an already placed pillar close to the centre of the hotspot made the placement at the centre of the hotspot unfeasible.

The second phase, called the analysis phase, is about analysing the results of the solution constructed in the construction phase. Thus, the heatmap resulting from the simulation with the pillar placement of the last iteration is considered. In the first iteration, this is the result of simulating the empty pillarmap.

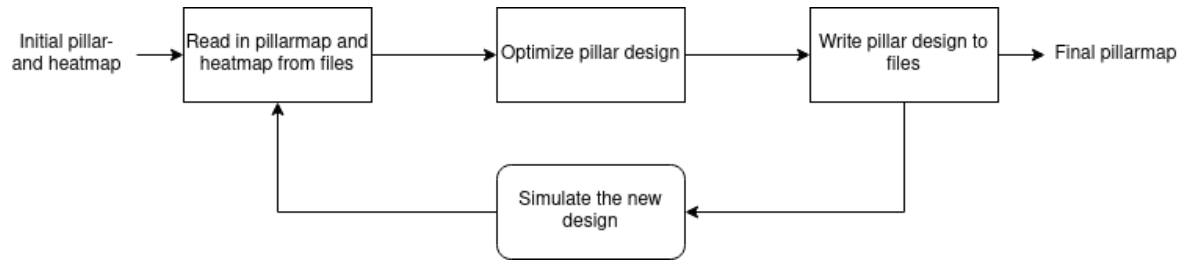


Figure 3.5: Simulation flow

Blame should be assigned to poor-performing elements, this could be the pillars which are positioned in the worst places. Thus, places where the temperature is the lowest.
In the prioritisation phase of the iteration, which is the final stage, blame assigned in the previous phase should be used to move elements towards the front of the sequence. In the context of this project, the ordering is made based on the temperature of the pillars. The pillars with the lowest temperatures are addressed first.
In this version of the strategy, pillars are placed one by one and not altered after placing, they will be annotated as restricted. This also means that the prioritisation phase can be skipped. The analysis phase can be used to read in the resulting heatmap so the construction phase can use it to optimally place new pillars. To move pillars to their optimal location, namely the feasible location which has the highest temperature, the same strategy as the stochastic hill climber algorithm is used. However, this is done for one pillar, without moving the existing pillars.
In practice first, an empty list for the pillars is created. Then, loop for the maximum number of pillars. In this loop, a dummy pillar is created at a random location. This pillar is added to the list of pillars and the pillars and feasibility map are updated by running the stochastic hill climber algorithm. Note that only the last placed pillar is moved since this is the only dummy pillar. Next, the pillar is updated and is now restricted. Then the current pillarmap is simulated on the simulation setup and the heatmap used by the stochastic hill climber is updated. When the maximum amount of pillars is placed, the list of pillars and feasibility map is returned.
This results in the following algorithm:

Algorithm 5: Squeaky wheel optimisation

procedure squeaky_wheel_optimisation(dimensions, feasibility_map, optimisation_steps, input_step_size, min_distance)
  $pillars \leftarrow []$
  for $maximum\_pillars$ **do**
    $pillar \leftarrow generate\_random\_pillar(dimensions, feasibility\_map)$
    $pillars.add(pillar)$
    $pillars, feasibility\_map \leftarrow stochastic\_hill\_climber(pillars, dimensions, feasibility\_map, optimisation\_steps, input\_step\_size, min\_distance)$
    $pillar.kind \leftarrow restricted$
    $run\_simulation(pillars)$
  return $(pillars, feasibility\_map)$
end procedure

3.5. Stochastic Hill Climber combined with Squeaky Wheel Optimisation

Both strategies in the sections above have their advantages and disadvantages, by combining them a different strategy can be created. The stochastic hill climber moves all pillars towards the hotspots, but because it does not re-simulate, a lot of the pillars are moved in the direction of the hotspots but are not necessarily placed well. Because the stochastic hill climber keeps trying to move all the pillars, iteration for iteration squeezes as many pillars as possible below the hotspots. On the other hand, the squeaky

wheel optimisation approach places the pillars in the centre of the hotspot, but because it cannot move a pillar once it is placed, the density below hotspots is much lower than for the stochastic hill climber. Because the squeaky wheel optimisation uses re-simulation, it is also able to address trouble spots outside of the original hotspots and distribute the heat more evenly.
These strategies can be combined by first using the stochastic hill climber to place as many pillars as possible below the hotspots. This should help with bringing the maximum temperature of the chip down. Then, this can be improved by analysing the resulting heatmap, blaming the pillar which has the lowest temperature and then moving the pillar to a better location. After that, re-simulate again and repeat the process. This should ensure that the heat on the chip is distributed better as a whole, and it should make the overall heat distribution more uniform.
In practice, the algorithm starts by creating an empty list to store the pillars. After that, the algorithm loops for the maximum number of pillars. In this loop, a random pillar is created of the kind dummy. This pillar is added to the list of pillars. Then the location of this pillar is optimised by running the stochastic hill climber algorithm again. Then this pillar is set to be restricted and cannot be altered. After the loop finishes, a new loop starts which loops for the predetermined number of squeaky wheel optimisation steps. In this loop, the pillar with the lowest temperature is first removed. After that, the squeaky wheel optimisation algorithm is used to place a new pillar. When his loop is finished, the pillars and feasibility map are returned. his results in the following pseudocode:

Algorithm 6: Stochastic Hill Climber combined with Squeaky Wheel Optimisation

```
procedure shc_swo_combined(dimensions, feasibility_map, optimisation_steps, SWO_steps,
input_step_size, min_distance)
    pillars ← []
    for maximum_pillars do
        pillar ← generate_random_pillar(dimensions, feasibility_map)
        pillars.add(pillar)
    pillars, feasibility_map ← stochastic_hill_climber(pillars, dimensions,
    feasibility_map, optimisation_steps, input_step_size, min_distance)
    pillars.kind ← restricted
    for SWO_steps do
        remove_lowest_temperature_pillar(pillars, feasibility_map)
        pillars, feasibility_map ← squeaky_wheel_optimisation(dimensions, feasibility_map,
        optimisation_steps, input_step_size, min_distance)
    return (pillars, feasibility_map)
end procedure
```

4

Experimental Design

First, the objectives of the experimental design are discussed. Then, the architecture of different test setups is considered, and the tools in which these setups are created and simulated. After that, the overall simulation and optimisation flow are considered, also paying attention to the interaction between code and tools. After that, it is verified whether the tools and setups yield the expected output results. Finally, the hardware which is used to run these experiments on is discussed.

4.1. Objectives

The main objective of the experimental design is to produce results of the different optimisation strategies in different scenarios that can be used to answer the research questions. These results can then be compared to each other to evaluate how effective the strategies are at distributing the heat over the chip. It is important to have a test setup where it can be observed how the changes in the underlying pillar map influence the results in terms of thermal heat distribution. On top of that, testing the strategies in an actual product setup makes it possible to evaluate how the strategies would work in practice.

4.2. Experimental setup

The experiments are set up in a tool called Ansys RedHawk-SC ElectroThermal [1], this tool is also used to do the simulation and verification. Ansys Redhawk-SC Electrothermal is a multiphysics solution for analysing power integrity, signal integrity, thermal behaviour and thermo-mechanical stress in multi-die chip packages and interconnects.
In the following section, the test setup is considered, this setup is inspired by an actual product but is more abstract. Then, the product is discussed, also discussing changes made to the setup to make it more suitable for testing and simulations.
The amount of different chips that can be used for testing is limited to two. One is a test setup specifically created for the experiments, the other is called IPTV and is an actual product. On the test setup, two different scenarios can be evaluated. On the actual product only one scenario can be evaluated. While testing on a total of three different scenarios limits the scope of this project, these scenarios are not case-specific and the results obtained should be generally applicable.

4.2.1. Test Setup

The test setup is a test vehicle which shares a lot of aspects with real products. On the top, a die is placed which generates heat and is powered according to a powermap. Then, copper pillars connect it to the substrate. The substrate is then connected to the PCB by a ball grid array. The die, substrate and PCB do not have complex shapes but are ordinary squares.
The original placement of the pillars is a grid placement of 22x18 pillars with a diameter of 0.09 mm, resulting in the placement of 396 pillars with a distance of 0.2324mm between them in the X-direction and 0.2276mm in the Y-direction. This placement is comparable to the placement of the intellectual property test vehicle (IPTV) described in the next section and line with the placement rules of section

[1] https://www.ansys.com/products/semiconductors/ansys-redhawk-sc-electrothermal

Components	Dimensions LxWxH(mm)
Die	5x3.99x0.3
Underfill	5x3.99x0.07
Laminate	10x10x0.52
BGA	10x10x0.3
PCB	100x100x1.694

Table 4.1: Experimental Setup Table

2.3.3.

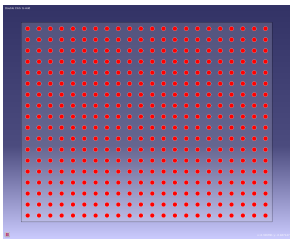


Figure 4.1: Standard grid layout

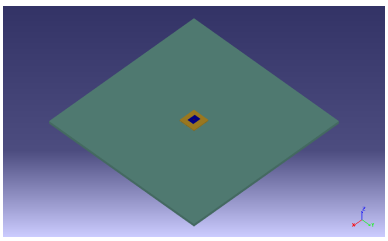


Figure 4.2: Test Setup Overview - SE

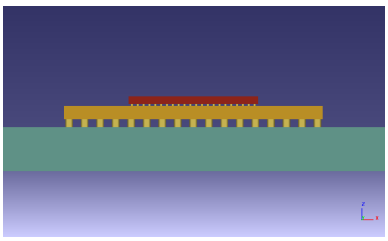


Figure 4.3: Test Setup Overview - Front

The test setup has two different powermaps; the single hotspot powermap, which has one spot of extensive power usage in the middle of the die, and the dual hotspot powermap which has a spot of extensive power usage in the middle of the die, as well as a spot in the top-right corner.

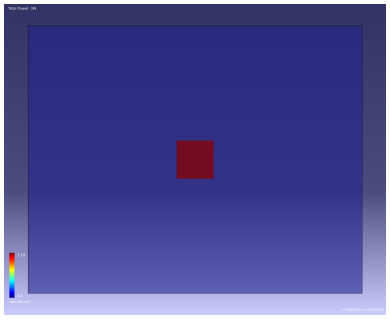


Figure 4.4: Test setup dual hotspot Powermap

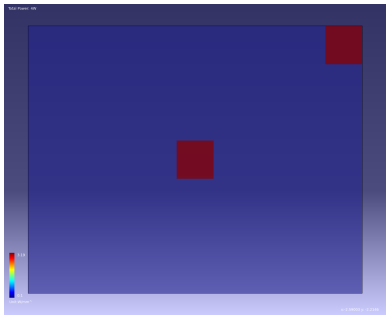


Figure 4.5: Test setup dual hotspot Powermap

4.2.2. Actual Product Setup

For the actual product setup, an actual product of NXP Semiconductors is used, but in this work, the product is called the intellectual property test vehicle (IPTV). The vehicle has a complex structure, it has a two-layer laminate design and has lots of complex shapes and connections on the die itself. This can be a problem if one of the pillars is moved to a theoretically better position, but it happens to be on a non-conductive shape of the substrate, it is possible to interpret the temperature change the wrong way. Therefore, to make sure changes in temperature are due to the topographical placements of the pillars, the complex structures on the substrate are removed in the experiments and replaced by homogenous structures.

The default pillar placement of IPTV has been constructed by placing pillars for routing, removing them from certain area's and finally laying a grid over it. As a result, the pillarmap can be divided into multiple sections; sections where no pillars should be placed at all, sections where the pillars should stay as they are placed in the original design and finally a big section where their place can be altered. In the experiments, the optimisation strategies are only allowed to move pillars in the latest section and are not allowed to place them back in one of the other sections. As a result, a realistic pillarmap should be

obtained.
The powermap of IPTV is also complex. It is the result of an actual operating scenario provided by NXP Semiconductors. Because of the complex structure of IPTV and the complex powermap, the resulting heatmap is also complex.
If supplied, the confidential supplement presents an overview of the actual product setup together with the powermap. After that, the standard pillar placement is shown, as well as the pillar placement of all restricted pillars. This restricted pillarmap shows the pillars which cannot be moved and have to stay in the same position during the experiments.

4.2.3. Simulation

While Ansys Redhawk-ET is a great tool to use by hand, automatically simulating and retrieving results is less trivial. The process of running these simulations was not known beforehand and it is not described in any documentation. The simulation software is not able to understand Python instructions and does not have an Application Programming Interface (API). To simulate the original and new pillar maps, the code which alters the pillar placement is executed on the same machine as the machine which can start the simulations. First, the Python code comes up with a new pillarmap and writes the new map in the right format to the right file. This file will be used by the simulation software once it is started, but it has to be altered before the tool starts.
The tool can be given instructions by supplying a TCL file when the tool is started by the command line, this is done by the Python script, which halts until the simulation software is done and closed. The TCL file is executed on startup. When the simulation instructions together with the 'exit' keyword are supplied in the TCL file, the tool exits immediately after the last simulation is finished. After that, the results are saved in a heat file in a specific directory in txt format.
Then, as the Python code does not halt anymore, since the simulation is finished, it can read the heat file from the right directory. After reading the new heatmap, and altering the pillar placement accordingly, the code can start the simulation again.
An important setting, which influences both the results directly as well as how the results are written into files, is the meshing. Meshing is done by placing pitches of 30 microns on the die. This results in 22512 data points on the die of the test setup and 57807 on the die of the actual product setup. On top of that, the tool also meshes complex structures into multiple points, it for example creates 4 points for every pillar. When the simulations are finished, the x,y, and z-coordinates and the final temperature on these positions are written to a file. Therefore, for the analysis, it is important to keep the meshing constant.

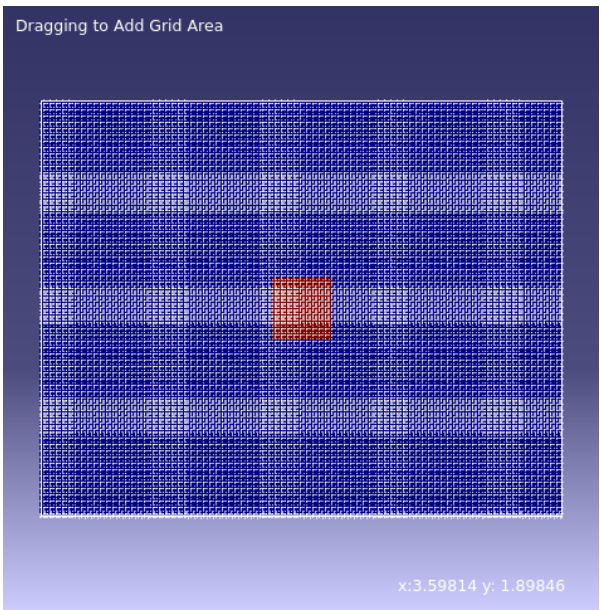


Figure 4.6: Test Setup Meshing

For all simulations, the simulations settings are the following:

- Multiphysics type: Thermal
- Simulation type: Static
- Boundary Type: THETA-JA
- Ambient temperature (C): 50
- Air speed (m/s): 0
- Solver type: Auto
- Power Ratio 0.05
- Max iteration: 10
- Temperature Diff(C): 0.1
- Geometry margin 0.1 um
- Maximum Edge Length in mesh: 10000 um
- Triangle Quality Control: 100 um
- Minimal Angle(deg): 15
- Advanced mesh control: Line
- Dielectric Filling Expanding(%): 5

4.2.4. Runtime

The runtime varies a little bit for each simulation, but over time the differences are minimal.
The test setup simulations take around two minutes to complete when run individually. However, when multiple simulations are done at the same time, for instance, fifty, it takes around ten minutes.
Because the actual product setup is more complex, it also takes around twelve minutes to complete a single simulation, when multiple simulations are done it can take up to twenty minutes.

4.3. Hardware

The Python scripts are run on a RedHat remote desktop with two cores and 16GB memory, provided by NXP Semiconductors. The simulation software is run on a cluster which has nodes with up to 71 cores and 1TB memory, also made available by NXP Semiconductors.

5 Results

In this chapter the results of the experiments are displayed and discussed. The different optimisation strategies are compared to the default way of placing pillars. The optimisation strategies are also compared to the lower- and upper boundaries of the setups in terms of heat. These boundaries are obtained by running the different experimental setups with a complete plate of copper instead of a layer of pillars and without any pillars. Replacing the pillars with a full plate of copper should conduct the heat most optimally. These scenarios should help us to place the obtained heat gradients in perspective. Firstly, the results of the test setup are considered, starting with a single hotspot and after that multiple hotspots. Then, results of the optimisation strategies on intellectual property test vehicles are discussed, using actual product power usages, and thus actual product heatmaps.
The influence of the pillar diameter on the success of the optimisation strategies is also evaluated, this is done on the test setup with the dual hotspot powermap.
The strategies are evaluated on multiple benchmarks, such as the maximum temperature on the chip, average temperature on the chip and how the heat is distributed. While the runtimes of the different algorithms are considered, this is not the main concern. The process of designing a chip takes a long time and therefore even if the experiments take a few days, it should not be a problem.

5.1. Test Setup

The results in this section are obtained with the experimental setup described in section 4.2.1. While this may not be an actual product setup, it still shares a lot of characteristics with product setups and therefore it is expected this environment yields significant results. Also, because the lower layers of this setup do not add any complexity, all obtained deltas in the results should solely be a result of the placement of pillars.

5.1.1. Single Hotspot Powermap

Firstly, the results of the single hotspot powermap are evaluated. This powermap only has a single place on the die where extensive power is used, namely the middle of the chip. As a result of the higher power usage in the middle of the chip, in the resulting heatmap, a hotspot is expected in the middle of the chip. Outside of this hotspot a uniform distribution of heat is expected, this is observable in the following heatmaps:

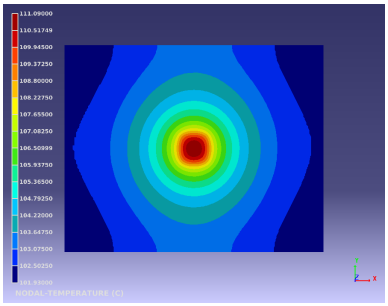


Figure 5.1: Heatmap - no pillars single hotspot powermap

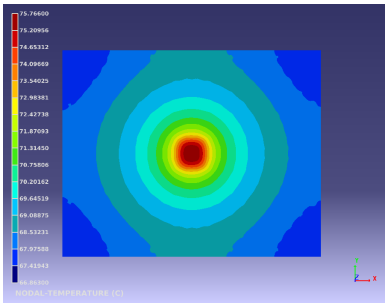


Figure 5.2: Heatmap - Grid placement single hotspot powermap

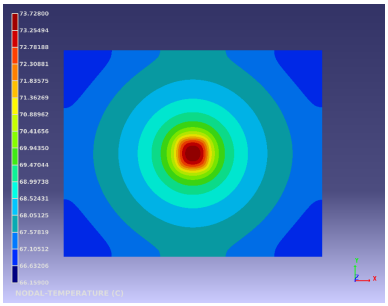


Figure 5.3: Heatmap - Full copper single hotspot powermap

From the figures above it becomes clear what the lower boundary, default case and the upper boundary are in terms of heat distribution. In the simulation without any pillars, the maximum temperature of 111.09 degrees Celsius was measured, this is the upper bound. On the other hand, where the layer of pillars is replaced with a plate of copper, the maximum temperature measured is only 73.73 degrees Celsius, which is the lower bound. The grid placement of figure 4.1 is a good benchmark for the different optimisation strategies to be measured against, the maximum temperature measured in this simulation is 75.77 degrees Celsius. Thus the standard placement can be improved with at most 2.04 degrees Celsius or 2.7%.
After applying the three different optimisation strategies, the following three pillar layouts were obtained:

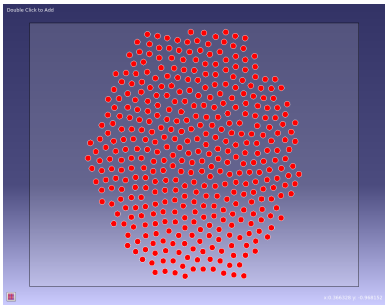


Figure 5.4: Layout - Stochastic Hill climber single hotspot powermap

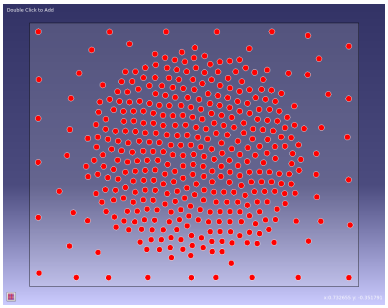


Figure 5.5: Layout - Squeaky wheel optimisation single hotspot powermap

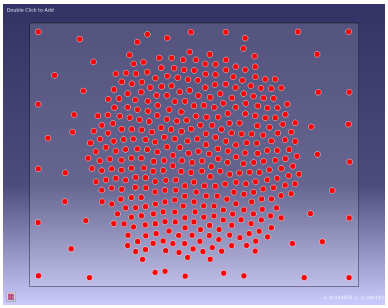


Figure 5.6: Layout - Stochastic Hill Climber SWO single hotspot powermap

It can be observed that in figure 5.4 the hill climbing algorithm only considers the heatmap from figure 5.1. The middle image, figure 5.5, has the pillars more spread out, a result of the re-simulation. It is also quite clear that the resulting layout of figure 5.6 is obtained by optimising the layout shown in figure 5.4.
The simulation results corresponding to the different layouts are shown below.

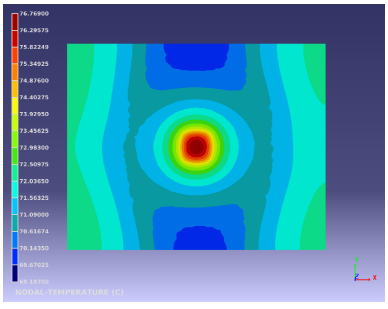


Figure 5.7: Heatmap - Hill climber single hotspot powermap

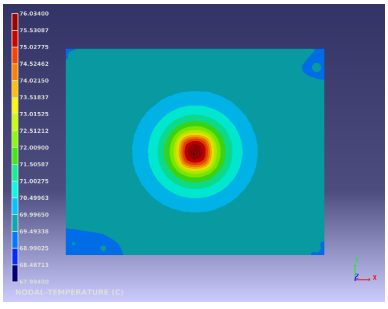


Figure 5.8: Heatmap - Squeaky wheel optimisation uniform single hotspot

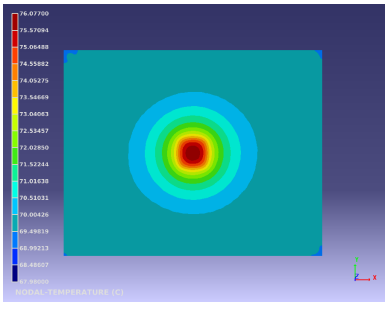


Figure 5.9: Heatmap - Hill Climber SWO single hotspot powermap

The first thing that comes to mind when looking at figure 5.7 is that there is a non-uniform distribution of temperature outside of the hotspot. On the top and bottom sides of the chip, there are two big sections of blue, thus relatively cold sections. Also, there is a very big difference between the coldest and the hottest points in terms of temperature. The lowest temperature measured on the die is 69.18 degrees Celsius, while the highest temperature measured is 76.77 degrees Celsius. This is 1 degree, or 1.3% higher than the highest temperature measured for the default placement.
For the second strategy, squeaky wheel optimisation, there is a much more even spread of heat across the die, which can be observed in figure 5.8. The points outside the hotspot have a comparable temperature with a few small fluctuations at the corners. However, the highest temperature measured on the chip is 76.03 degrees Celsius, which is 0.23 degrees Celsius higher than the highest temperatures measured in the grid placement. This is an increase in temperature of 0.3%.
Figure 5.9 shows the heatmap of the last strategy, starting with the hill climber and optimising it using the squeaky wheel strategy. This figure also shows a homogeneous heat distribution with the points outside of the hotspot all having a comparable temperature. The highest temperature measured on the chip for this strategy was 76.08 degrees Celcius, which is comparable to the squeaky wheel strategy but still an increase of 0.4% compared to the grid placement strategy.
The figures above show a very clear overview of how the heat is distributed on the die. Also, because there is some randomisation involved and measurements of the simulation tool can fluctuate by around a hundredth of a degree. The strategies are compared to each other by averaging the results over 50 runs, while also calculating the standard error. The grid placement has the same pillar map for all simulations, differences in results between runs for this strategy are therefore the result of the simulation tool's fluctuations.
In the following figures, the results are plotted by taking the highest on-chip temperature for each X-coordinate in one of the figures and doing the same for each Y-coordinate in the other figure. This should give a good representation of the overall heat distributions of the different strategies, but more importantly, the height of the hotspots of the different strategies. Also, two times the standard error is plotted in a band around the average of the 50 runs to show the consistency of the different strategies.

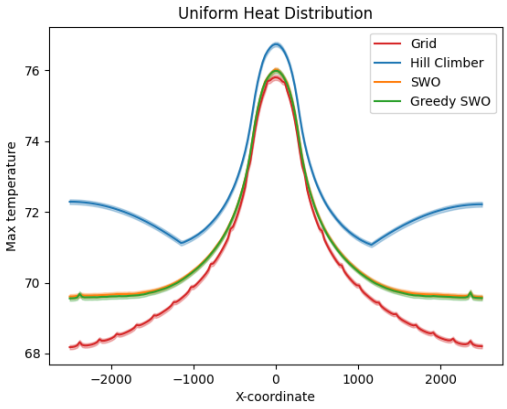


Figure 5.10: Maximum heat per X-coordinate single hotspot powermap

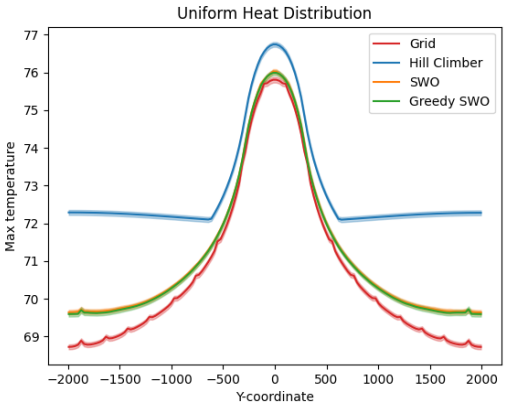


Figure 5.11: Maximum heat per Y-coordinate single hotspot powermap

Figures 5.10 and 5.11 clearly show the hotspots as observed in the earlier shown heatmaps. These plots also clearly show that the hill climbing strategy produces the highest temperature on each X and/or Y coordinate line. In the middle section of the graphs, where the hotspot is located, the grid strategy performs the best, having the lowest temperatures. Followed by the hill climber SWO combination and then the squeaky wheel optimisation. However, these differences are subtle and not very significant. This is in line with the measurements on the heatmaps earlier in this section, where the temperature differences were around 0.3 degrees Celsius. The further off-centre, the higher the differences between the three optimisation strategies. This is also shown by the heatmaps. However, there is no significant difference over the whole graph between the squeaky wheel optimisation strategy and the hill climber combined with the squeaky wheel optimisation strategy.

5.1.2. Dual Hotspot Powermap

The second scenario considered has two hotspots. One hotspot is located at the centre of the chip and one is located in the top-right corner. Consequently, a non-uniform heatmap with 2 hotspots is expected. In the following figures the resulting heatmaps are shown, again the scenario without any pillars, the default grid strategy and the scenario where the whole layer is a plate of copper:

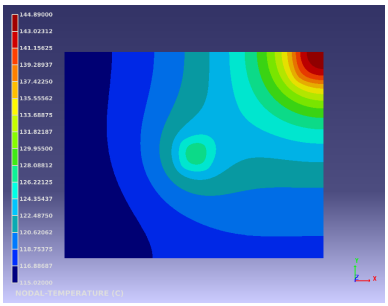


Figure 5.12: Heatmap - no pillars dual hotspot powermap

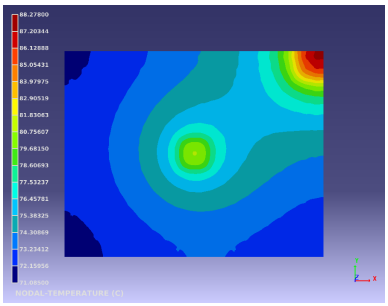


Figure 5.13: Heatmap - Grid placement dual hotspot powermap

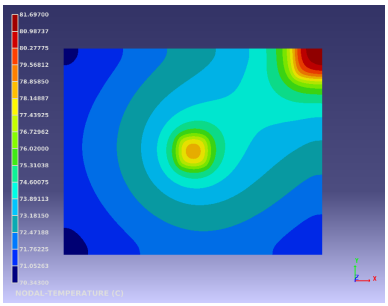


Figure 5.14: Heatmap - Full copper dual hotspot powermap

It can be observed that all the heatmaps above have a non-uniform heat distribution outside of the hotspots on the chip. Also, all three of them have a small hotspot in the middle and a much 'higher' hotspot in the top-right corner of the die.
When all pillars are removed, resulting in the heatmap of figure 5.12, the temperature rises to a maximum height of 144.89 degrees Celsius. This is much higher than the maximum temperature of 88.28 degrees Celsius measured on the heat map of the grid placement of figure 4.1. The heatmap obtained by simulating the grid placement is shown in figure 5.13. The heatmap shown in figure 5.14, the result of replacing the whole layer with a plate of copper, has a maximum measured temperature of 81.70 degrees Celsius. Replacing the layer with a plate of copper reduces the maximum temperature by 7.45% compared to the grid placement.
After running the different optimisation strategies, we obtain the following pillar layouts:

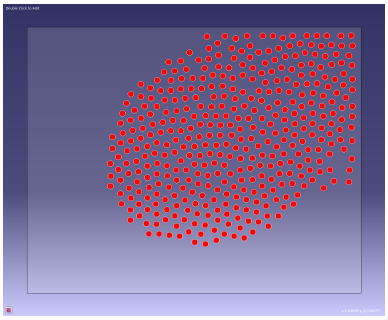


Figure 5.15: Layout - Stochastic Hill climber dual hotspot powermap

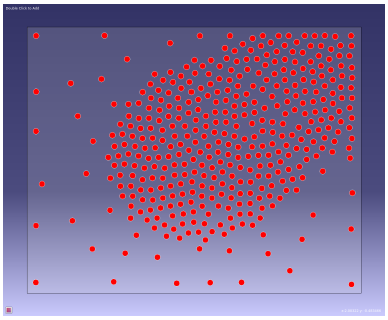


Figure 5.16: Layout - Squeaky wheel optimisation dual hotspot powermap

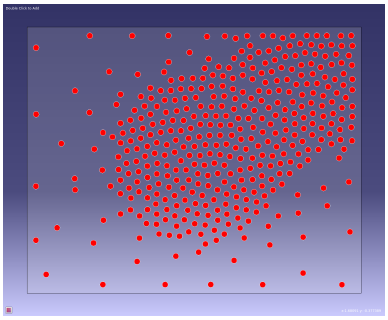


Figure 5.17: Layout - Stochastic Hill Climber SWO dual hotspot powermap

The resulting pillar layout presented in figure 5.15 shows a very dense map where all pillars are very close to each other and clubbed around the hotspots. This is a result of the hill climbing strategy where the algorithm tries to move the pillars individually to the hottest points without re-simulating the placement. Because the squeaky wheel optimisation strategy does use re-simulation, the pillars are much more spread out, this can be observed in figure 5.16. Finally, in figure 5.17 the combination of the other two figures can be seen. It has very dense areas under the hotspots, but there are still pillars spread around the whole surface of the layer.
The heatmaps corresponding to the above layouts are also obtained:

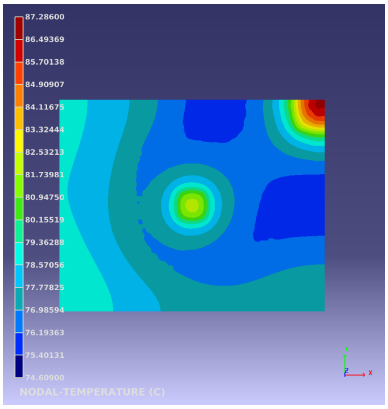


Figure 5.18: Heatmap - Hill climber dual hotspot powermap

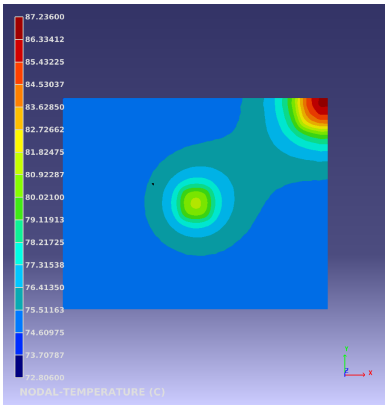


Figure 5.19: Heatmap - Squeaky wheel optimisation dual hotspot powermap

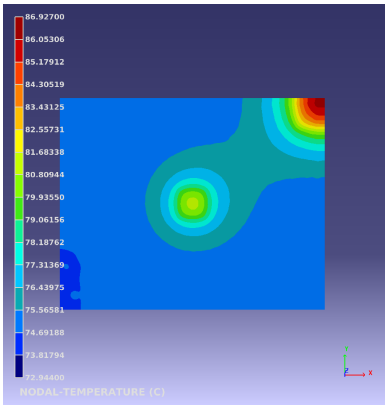


Figure 5.20: Heatmap - Hill Climber SWO dual hotspot powermap

While the heat distribution of figure 5.18 still shows two hotspots, the maximum measured temperature is 87.29 degrees Celcius. This is around a whole degree lower than the maximum measured temperature in the default case and reduces the maximum temperature of the current placement by 1.12%. The other two figures; 5.19 and 5.20, show a more uniform thermal heat distribution, where the points outside of the hotspots have an equal temperature. The hottest point measured when the squeaky wheel optimisation was used had a temperature of 87.24 degrees Celsius. This is a degree lower than the default case and reduces the maximum temperature by 1.17% compared to the default case. The final strategy, the combination of the hill climber and squeaky wheel optimisation, had a maximum measured temperature of 86.93 degrees Celsius, thus the lowest of all cases. The final strategy reduces the maximum temperature on the chip by 1.53% compared to the default case.
When the results are again averaged over multiple runs, and compared to each other by taking the maximum temperature measured for each X-coordinate in one figure and each Y-coordinate in another figure, we get the following two figures:

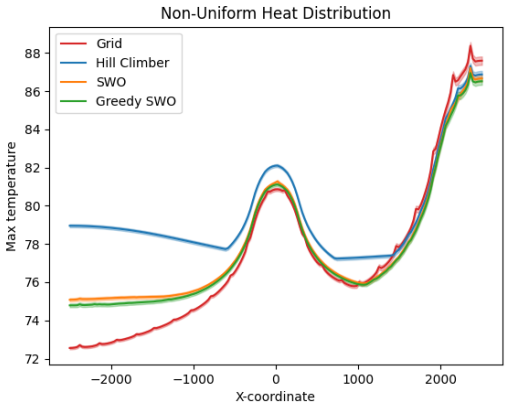


Figure 5.21: Maximum heat per X-coordinate dual hotspot powermap

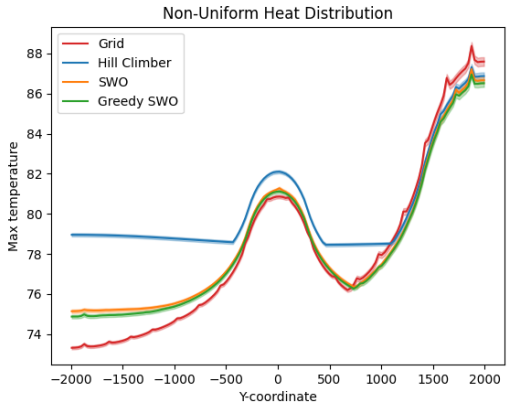


Figure 5.22: Maximum heat per Y-coordinate dual hotspot powermap

Figures 5.21 and 5.22 look comparable, which is expected since the hotspots are in the middle and top-right, so looking from the X-side or Y-side should not change the perspective too much.
It can be observed that the highest points in the graph are all on the right-hand side, this corresponds with the heatmaps shown earlier in this section, where the hottest points have always been in the top-right corner of the chip. It is also clear that the line represented by the grid strategy is significantly higher than the other lines in this section, which corresponds with the delta observed in the heatmaps earlier in this section. Among the three different strategies, the differences are more subtle and less significant. The results show that the highest temperature among the optimisation strategies is obtained by the hill climber strategy, followed by the squeaky wheel optimisation and then the combination of the hill

climber and squeaky wheel optimisation. However, in the middle section of the graph, the grid strategy performs better and has a lower maximum temperature for different X and Y coordinates compared to the optimisation strategies. This corresponds with the results of subsection 5.1.1 where a comparable hotspot was considered. Among the three different optimisation strategies, the hill climber performs the worst, obtaining the highest temperatures in the middle section, followed by again the squeaky optimisation and finally the combination of the two.
Then, on the left-hand side of both graphs, which represents the left-bottom part of the chip, the temperatures measured during simulation show clearly that the grid temperatures are the lowest, followed by the combination of the hill climber and squeaky wheel optimisation, then the squeaky wheel optimisation and finally, having the highest temperatures, the hill climber.

Alternative pillar diameters
In this part of the paragraph, the influence of pillar diameter when running the Squeaky Wheel Optimisation on the dual hotspot powermap is evaluated. The number of pillars used is constant, namely 396. Although larger pillars result in a higher copper coverage at equal pillar count, the results are still interesting. The manufacturing rules require pillars with a bigger diameter to have a greater minimal distance between each of the pillars. The results are averaged over 10 runs and plotted in the same manner as before, only plotting the maximum temperature point for each X-Coordinate and Y-Coordinate. The following plots were obtained:

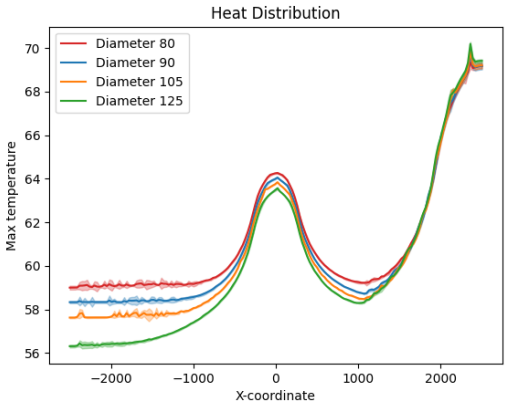


Figure 5.23: SWO different diameters maximum temperature dual hotspot powermap

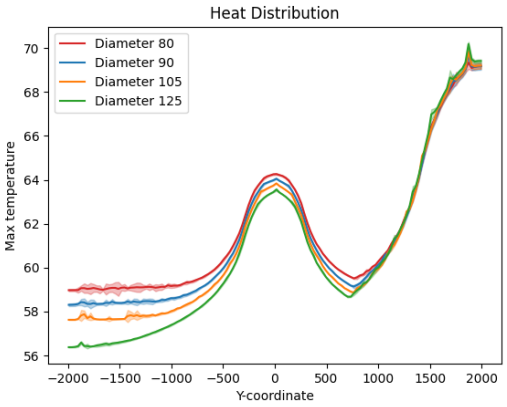


Figure 5.24: SWO different diameters average temperature dual hotspot powermap

The plots show that an increased pillar diameter helps with bringing the average temperature down, this is mainly visible in the left tail of the graph and the section around X-coordinate 1000. Also, the maximum temperatures at the middle hotspot show that the increased amount of copper of the pillars with a larger diameter reduces the overall temperature and therewith brings the hotspot down. The opposite is true for the hotspot in the top-right of the chip, this can be observed at the right tail of the plots. Here, the pillars with a smaller diameter can reduce the maximum temperature more effectively.

5.2. IPTV

In this section, the results of the experiments with the intellectual property test vehicle are considered. The figures which contain the powermap, pillarmaps and heatmaps can be found in the confidential supplement if supplied.
First, the vehicle without pillars, the original pillarmap is better known as grid placement and the scenario where the layer of pillars is replaced by a full layer of copper is run. The hotspot is for all scenarios located at the same spot, in the top-left, this corresponds with the higher power density in this region. The heatmap with no pillars gets as hot as 87.55 degrees Celsius, the heat cannot be conducted efficiently to the lower layers. In the original placement, the grid placement, the maximum temperature on the chip gets as high as 69.88 degrees Celsius. This will be used as the benchmark for the optimisation strategies. Finally, the scenario with the full layer of copper should be the most optimal one and the maximum temperature is 65.62 degrees Celsius. Note that this is not a realistic solution but merely a

reference. Apart from the maximum temperature, the hotspot generated by the scenario without any pillars covers a larger area in comparison to the hotspots of the other scenarios. This is a clear sign of heat also spreading. On top of that, the temperature difference between the default grid placement and the full layer of copper is 4.26 degrees Celsius.
The pillar layout obtained by the Stochastic Hill Climber is very similar to that of the Stochastic Hill Climber combined with the Squeaky Wheel Optimisation. This is because the Hill Climber formed the starting point for the combination of strategies. The resulting pillarmap of the Squeaky Wheel Optimisation shows a clustering of pillars around the hotspots. The pillars placed by the Squeaky Wheel Optimisation in the central areas of the chip are more spread out compared to the maps generated by the other strategies.
The heatmaps obtained by simulating the resulting pillarmaps of the optimisation strategies show a similar pattern as the standard heatmaps obtained earlier. The big hotspot at the top-left still dominates all other hotspots, and there are two cold spots on the bottom-left and top-right side of the chip. The maximum temperatures are different to the maximum temperature of 69.88 degrees Celsius on the standard heatmap. The maximum measured temperature is 69.94 degrees Celsius for the Stochastic Hill Climber, thus very comparable to the maximum temperature of the default placement. The maximum temperature resulting from the Squeaky Wheel Optimisation is an increase in maximum temperature compared to the original placement of 0.02 degrees Celsius to 69.90 degrees Celsius. The final strategy, Stochastic Hill Climber combined with the Squeaky Wheel Optimisation, has a maximum temperature measured of 69.94 degrees Celcius. This is both very comparable to the result of the standard placement as well as to the results obtained by the other two optimisation strategies.
In the next two plots, the results of the different optimisation strategies are compared to each other over the whole heatmap. Again, for each of the X-Coordinates and Y-Coordinates the maximum temperature is taken. The results are averaged over 10 runs and the standard error is again shaded over the lines. The following plots were obtained:

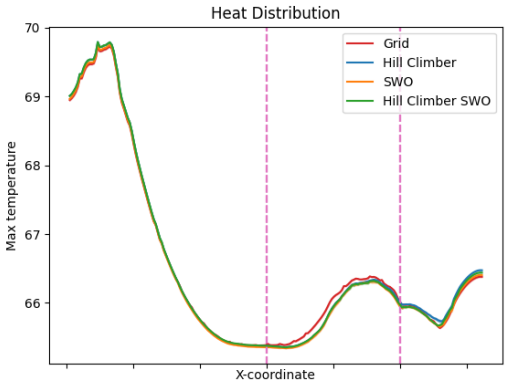


Figure 5.25: Maximum heat per X-coordinate IPTV

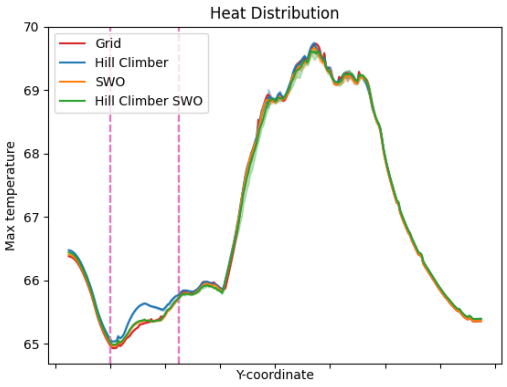


Figure 5.26: Maximum heat per Y-coordinate IPTV

The Stochastic Hill Climber, Squeaky Wheel Optimisation and the Stochastic Hill Climber combined with the Squeaky Wheel Optimisation are very comparable to the grid placement in hot sections of the chip. However, this can be expected because the algorithms were not allowed to alter these regions. In figure 5.25 it can be observed that in other regions, especially in the region between the two dotted pink lines, the optimisation techniques can bring down the maximum heat measured for these X-Coordinates. This reduction is limited to a few tenths of a degree. In this region, the algorithms had the freedom to change the pillarmap, but there was no big hotspot there. In the other plot of figure 5.26 these differences are less observable and the strategies perform comparable, but the Hill Climber performs significantly worse on the section enclosed by the two pink dotted lines.

5.3. Runtime

The runtime of the algorithms depends mainly on the number of pillars which needs to be placed, certain parameters and the time it takes to do the simulations. The Stochastic Hill Climber does not use simulation, and therefore the runtime only depends on the number of pillars which are placed and the number of optimisation steps used. The runtime for the Stochastic Hill Climber scales linearly with the number of pillars that need to be placed and the optimisation steps used. The runs performed where 396 pillars were placed with 20000 optimisation steps, finished around an hour. The runs for IPTV, where 810 pillars were placed and again 20000 optimisation steps were used finished around two hours.

The runtime of the Squeaky Wheel Optimisation is dominated by the time it takes to run the simulations. Placing a pillar in an optimal location takes constant time, but running the simulation can take around 8 minutes for the small map but 15 minutes for the more complex map of IPTV. Therefore, placing a total of 396 pillars and running a simulation for all of them can result in a total runtime of 50 hours. When IPTV is considered the total runtime can exceed 100 hours.

The runtime of the combination of the Stochastic Hill Climber and Squeaky Wheel Optimisation is also dominated by the time it takes to do a simulation. Removing the worst placed pillar and placing a new one takes constant time. The simulation times are the same as mentioned before for the small and bigger maps. However, this strategy requires fewer simulations, this resulted in a simulation time of 20 hours for the smaller maps and 50 hours for IPTV.

6 Discussion

In this chapter the research questions are discussed and answered with the use of previous chapters. After that, the limitations of this work are considered. Lastly, it is discussed what future research can be done related to this work and how it can be shaped.

6.1. Research questions

- How can topographical pillar placement be thermally optimised with the use of Artificial Intelligence?

In this work it is shown that topographical pillar placement can be thermally optimised for chips with a non-uniform powermap using a hill climber, squeaky wheel optimisation or a combination of these strategies. These strategies can bring down the maximum temperature and make the heat distribution outside the hotspot more homogeneous. However, these strategies were only partly able to show the same results on an actual product setup and are not able to improve the pillar placement of the chips with a single central hotspot.

- How can hotspots on flip-chip design be reduced by altering the pillar placement?

Hotspots are greatly reduced by increasing the number of pillars under the hotspot and centring a pillar in the middle of the hotspot.

- How can a non-uniform heat distribution be improved to move towards a homogeneous case?

With a squeaky wheel optimisation strategy, where the setup is re-simulated every iteration, it is possible to make a non-uniform heat distribution more homogeneous.

- Under what design conditions is improving heat distribution possible?

Improving the heat distribution requires freedom in the change of location of the pillars. It is most effective when the pillars can be manipulated in sections right below the hotspots.

- What constraints make it impossible to improve topographical pillar placement?

Constraints such as the fixed pillar constraints and forbidden regions, as seen in practice on the setup described in section 4.2.2, make it harder to improve the topographical pillar placement. When a change in pillar locations is impossible, or when the freedom is limited, for example when there is only a small section which is not a forbidden region, it is nearly impossible to improve the pillar placement significantly.

- How does the size of the pillar influence the pillar optimisation strategy results?

In section 5.1.2 the results of different pillar placements shows that the difference in diameter influences the result, but that this is dependent on the powermap. The smaller diameter performed better when the dual hotspot powermap was used. This, while the total area of copper was smaller than the total area of copper of the pillars with larger diameters. The bigger diameter pillars were better to conduct the heat in general because of the increased copper coverage. A better evaluation would be to compare multiple pillar diameters while keeping the area of copper consistent.

6.2. Limitations

The main limitation is that some manufacturing rules were scrapped to make the scope of the project smaller and reduce the complexity of the project. As a result, the pillarmaps obtained by the different algorithms do not comply with all the rules and are therefore not manufacturable.
Other limitations are related to the actual product experiments. These experiments are only conducted on one test vehicle. Also, this device had a hotspot which greatly dominated all other hotspots and was in a section where the underlying pillar map could not be altered. Therefore the algorithm could not optimise the pillar placement for this hotspot. Also, only a single powermap with a corresponding heatmap was evaluated. If multiple scenarios were evaluated the optimisation strategies could have performed better.
Also, the evaluations of the different pillar diameters were limited only to a single optimisation technique. It is unclear how the Hill Climbing strategy could be influenced by different pillar diameters. On top of that, mixing different pillar sizes was also not attempted. This would have made the project too complex and was not within the scope of the project.
Finally, the simulations and results all depend on a single optimisation tool. It could be possible that certain assumptions made by the tool influence the result in a way which is not representative of reality.

6.3. Future research

Arguably, this work provides a solid foundation to continue the research for topographical pillar placements in flip-chip design. Future research could first focus on verifying the optimisation strategies described in this work, with different tools and on a different setup. After that, more manufacturing rules could be introduced to make it possible to manufacture the pillarmaps.
Also, it would be interesting to employ new optimisation strategies. The optimisation strategies described in section 3 depend heavily on the results from a heat simulation. More complex strategies, where more sophisticated ideas of artificial intelligence are used, could yield more effective strategies. Because the current techniques highly depend on the simulation tool, the possibility to speed up the process is limited. A strategy which learns from old designs and the heatmaps those designs produce using machine learning techniques could speed up the process.
Finally, the strategies in this work do not take the underlying structure of the package or the chip itself into account. A strategy which can combine the powermap with the underlying architecture of the package or chip could be able to come up with better strategies which do not necessarily need re-simulations. A strategy that depends on the properties of the materials used in the different layers, for example.

7

Conclusion

At the start of this project, there was no procedure in place to automatically run and test pillar placement strategies. After trying out multiple tools and ways to automate simulation, and manipulate different files to change the underlying pillar structure, it was possible to fully automate this process. As a result, thorough experiments were run to compare the different created pillar placement strategies accurately. The role of this automated process is essential to conducting long-term experiments and also makes it possible to automatically evaluate strategies and designs in future research.

The optimisation strategies do not yield significantly better pillar placements compared to the general grid placement on a chip with a single hotspot powermap, where there is a single hotspot in the middle of a chip. The results of section 5.1.1 indicate that the different optimisation strategies did not outperform the general grid strategy in terms of maximum temperature on the chip. However, the optimisation strategies which used re-simulation and the squeaky wheel optimisation show a more even distribution of the heat outside the hotspot. The Stochastic Hill Climbing strategy does the opposite, as can be observed in figure 5.7. The temperature differences between sections outside the hotspot increased.

For a chip where there are two hotspots, one in the middle and one in the corner of the chip, the optimisation strategies are able the generate pillar placements which are better than the general grid strategy. In section 5.1.2 the optimisation strategies; Stochastic Hill Climber, Squeaky Wheel Optimisation and the combination of both, all reduced the maximum temperature on the chip by more than 1.1% compared to the grid placement, while the upper bound was around 7.45%. This is mostly due to the difference in the temperatures in the hotspot which is located in the top-right. For the hotspot in the middle of the chip, the results were comparable to the results of section 5.1.1, where the single hotspot in the middle of the chip is considered. The strategies which used used re-simulation and the squeaky wheel optimisation again improved the heat distribution outside the hotspots.

The results indicate that for some places, placing smaller pillars can be more effective, but in general, a higher diameter pillar is better to conduct the heat. The experiments related to how the pillar diameter influences the optimisation strategy, the squeaky wheel optimisation to be precise, were not very elaborate but still yielded interesting insights. The results showed that the pillars with the lower diameters were better able to bring the maximum temperature down. On the other hand, the pillars with a higher diameter were more successful in bringing the average temperature and the temperature in the middle of the chip down, due to the extra copper.

The results shown in section 5.2 indicate that the optimisation strategies were not very effective in bringing the maximum temperature down on this particular product setup. Only one actual product setup was used with a single powermap and corresponding heatmap. This heatmap was mostly dominated by a hotspot outside the regions where the optimisation strategies were allowed to place pillars. The pillarmaps generated by the optimisation techniques show that a lot of the pillars were placed close to this big hotspot but the hotspot did not change as a result. When the analysis is focused on a section where the heatmaps changed as a result of the changes in the underlying pillarmap, these three strategies were all able to bring the temperature down by a few tenths of a degree.

While the runtime was not very important to the project, the evaluations of the runtime show that these should not negatively influence the time it takes to design a chip. Because these strategies do not require human intervention or oversight, while it is still able to provide designers with useful pillar maps, they should speed up the design process.

References

[1] Jack S Kilby. “Invention of the integrated circuit”. In: *IEEE Transactions on electron devices* 23.7 (1976), pp. 648–654.

[2] Gordon E Moore et al. *Cramming more components onto integrated circuits*. 1965.

[3] Our world in data. *Moore’s Law: The number of transistors per microprocessor*. [Online; accessed June 2, 2022. 2022. url: https://ourworldindata.org/grapher/transistors-per-microprocessor.

[4] Fatima Hameed Khan, Muhammad Adeel Pasha, and Shahid Masud. “Advancements in Microprocessor Architecture for Ubiquitous AI—An Overview on History, Evolution, and Upcoming Challenges in AI Implementation”. In: *Micromachines* 12.6 (2021). issn: 2072-666X. doi: 10.3390/mi12060665. url: https://www.mdpi.com/2072-666X/12/6/665.

[5] Tai-Ran Hsu. “Reliability in MEMS packaging”. In: *2006 IEEE International Reliability Physics Symposium Proceedings*. IEEE. 2006, pp. 398–402.

[6] Denver Desouza. “Thermal Analysis of Cu Pillars”. en. In: (2021). url: https://repository.tudelft.nl/islandora/object/uuid%3A03abf581-a53e-44fc-a8c5-21a6921f82a3 (visited on 02/27/2022).

[7] Austin Lancaster and Manish Keswani. “Integrated circuit packaging review with an emphasis on 3D packaging”. In: *Integration* 60 (2018), pp. 204–212.

[8] Masayoshi Esashi. “Wafer level packaging of MEMS”. In: *Journal of Micromechanics and Microengineering* 18.7 (2008), p. 073001.

[9] Philip Garrou. “Wafer level chip scale packaging (WL-CSP): An overview”. In: *IEEE Transactions on Advanced Packaging* 23.2 (2000), pp. 198–205.

[10] Paul A Totta et al. “Chip-to-package interconnections”. In: *Microelectronics Packaging Handbook*. Springer, 1997, pp. 129–283.

[11] Peter Elenius and Lee Levine. “Comparing flip-chip and wire-bond interconnection technologies”. In: *Chip Scale Review* 4.6 (2000).

[12] Th Krems et al. “Millimeter-wave performance of chip interconnections using wire bonding and flip chip”. In: *1996 IEEE MTT-S International Microwave Symposium Digest*. Vol. 1. IEEE. 1996, pp. 247–250.

[13] Zhuqing Zhang and CP Wong. “Recent advances in flip-chip underfill: materials, process, and reliability”. In: *IEEE transactions on advanced packaging* 27.3 (2004), pp. 515–524.

[14] A. Shakouri and Yan Zhang. “On-chip solid-state cooling for integrated circuits using thin-film microrefrigerators”. In: *IEEE Transactions on Components and Packaging Technologies* 28.1 (2005), pp. 65–69. doi: 10.1109/TCAPT.2005.843219.

[15] Tamara Bechtold, Evgenii B Rudnyi, and Jan G Korvink. “Dynamic electro-thermal simulation of microsystems—a review”. In: *Journal of micromechanics and microengineering* 15.11 (2005), R17.

[16] VA Drebushchak. “The peltier effect”. In: *Journal of Thermal Analysis and Calorimetry* 91.1 (2008), pp. 311–315.

[17] Jayashree Saxena et al. “A case study of IR-drop in structured at-speed testing”. In: *International Test Conference, 2003. Proceedings. ITC 2003.* IEEE Computer Society. 2003, pp. 1098–1098.

[18] Toan Dinh et al. “Thermoresistive effect for advanced thermal sensors: Fundamentals, design considerations, and applications”. In: *Journal of Microelectromechanical Systems* 26.5 (2017), pp. 966–986.

[19] Unai Alvarado, Guillermo Bistué, and Iñigo Adín. “Power Considerations in Analog RF CMOS Circuits”. In: *Low Power RF Circuit Design in Standard CMOS Technology*. Springer, 2011, pp. 11–24.

[20] NXP Semiconductors. “Thermal Analysis of Semiconductor Systems.” In: ().

[21] Harry JM Veendrick. *Nanometer CMOS ICs*. Springer, 2017.

[22] LT Yeh. “Review of heat transfer technologies in electronic equipment”. In: (1995).

[23] Xiangning Lu et al. “Thermal conduction analysis and characterization of solder bumps in flip chip package”. In: *Applied Thermal Engineering* 36 (2012), pp. 181–187.

[24] Jie Zou and Alexander Balandin. “Phonon heat conduction in a semiconductor nanowire”. In: *Journal of Applied Physics* 89.5 (2001), pp. 2932–2938.

[25] Latif M Jiji and Latif Menashi Jiji. *Heat convection*. Springer, 2006.

[26] EM Davis et al. “Solid logic technology: versatile, high-performance microelectronics”. In: *IBM Journal of Research and Development* 8.2 (1964), pp. 102–114.

[27] Wen Shan Tsai et al. “Generational changes of flip chip interconnection technology”. In: *2017 12th International Microsystems, Packaging, Assembly and Circuits Technology Conference (IMPACT)*. IEEE. 2017, pp. 306–310.

[28] John H Lau. “Recent advances and new trends in flip chip technology”. In: *Journal of Electronic Packaging* 138.3 (2016), p. 030802.

[29] Reijiro Shoji Kazuto Saitoh. *Bump Structure Bump Forming Method and Package Connecting Body*. U.S. Patent 6229220B1, May. 2001.

[30] Francisca Tung. *Pillar connections for semiconductor chips and method of manufacture*. U.S. Patent 6681982B2, Jun. 2003.

[31] Valery M. DubinSridhar BalakrishnanMark Bohr. *Designs and methods for conductive bumps*. U.S. Patent 9543261B2, Jan. 2007.

[32] Andrew Yeoh et al. “Copper die bumps (first level interconnect) and low-K dielectrics in 65nm high volume manufacturing”. In: *56th Electronic Components and Technology Conference 2006*. IEEE. 2006, 5–pp.

[33] Wei Koh, Barry Lin, and Johnson Tai. “Copper pillar bump technology progress overview”. In: *2011 12th International Conference on Electronic Packaging Technology and High Density Packaging*. IEEE. 2011, pp. 1–5.

[34] C. K. Chang. “TSMC-NXP Cu Pillar Interconnect Rules.” In: (2020).

[35] Yang Liu et al. “Thermal simulation of flexible LED package enhanced with copper pillars”. In: *Journal of Semiconductors* 36.6 (2015), p. 064011.

[36] Andrea Schaerf. “Local search techniques for large high school timetabling problems”. In: *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans* 29.4 (1999), pp. 368–377.

[37] Garrett Lewellen et al. “A hybrid traveling salesman problem-squeaky wheel optimization planner for earth observational scheduling”. In: (2017).

[38] David P Clements and DE Joslin. “Squeaky wheel optimization”. In: *Journal of artificial intelligence research* (1999).

[39] David S Johnson, Christos H Papadimitriou, and Mihalis Yannakakis. “How easy is local search?” In: *Journal of computer and system sciences* 37.1 (1988), pp. 79–100.

[40] Ari Juels and Martin Wattenberg. “Stochastic hillclimbing as a baseline method for evaluating genetic algorithms”. In: *Advances in Neural Information Processing Systems* 8 (1995).

[41] Ender Özcan, Burak Bilgin, and Emin Erkan Korkmaz. “Hill climbers and mutational heuristics in hyperheuristics”. In: *Parallel Problem Solving from Nature-PPSN IX*. Springer, 2006, pp. 202–211.

[42] David E Joslin and David P Clements. “Squeaky wheel optimization”. In: *Journal of Artificial Intelligence Research* 10 (1999), pp. 353–373.

[43] Michael Jünger, Gerhard Reinelt, and Giovanni Rinaldi. “The traveling salesman problem”. In: *Handbooks in operations research and management science* 7 (1995), pp. 225–330.