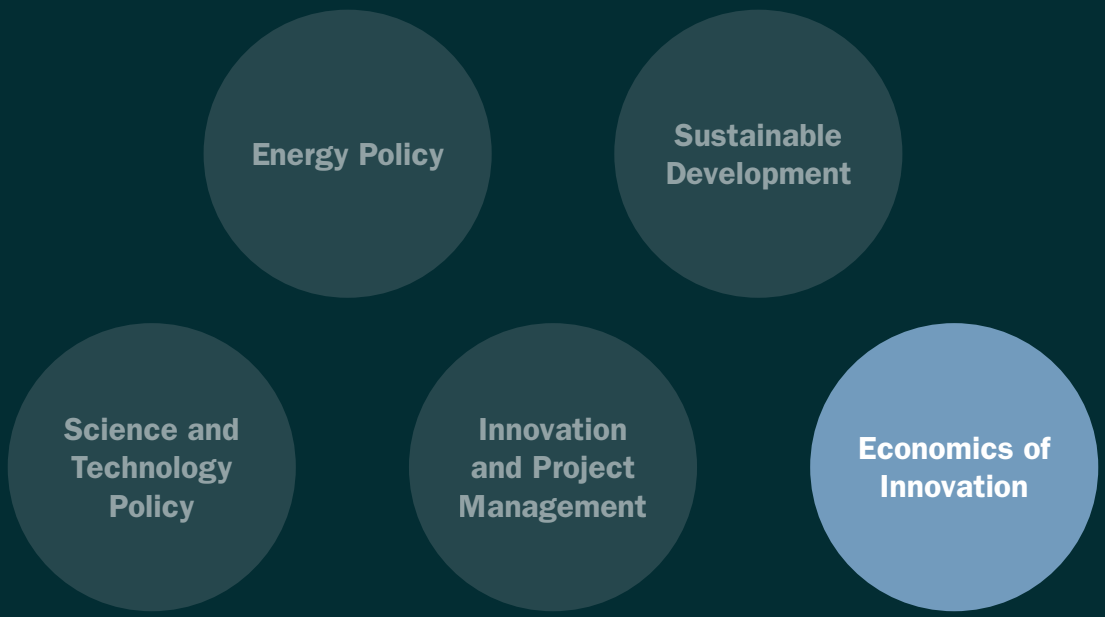
SCIENCE POLICY RESEARCH UNIT

SPRU Working Paper Series

SWPS 2020-01 (February)

On the Basis of Brain: Neural-Network-Inspired Change in General Purpose Chips

Ekaterina Prytkova and Simone Vannuccini



UNIVERSITY OF SUSSEX | BUSINESS SCHOOL

SPRU Working Paper Series (ISSN 2057-6668)

The SPRU Working Paper Series aims to accelerate the public availability of the research undertaken by SPRU-associated people, and other research that is of considerable interest within SPRU, providing access to early copies of SPRU research.

Guidelines for authors

Papers should be submitted to swps@sussex.ac.uk as a PDF or Word file. The first page should include: title, abstract, keywords, and authors' names and affiliations. The paper will be considered for publication by an Associate Editor, who may ask two referees to provide a light review. We aim to send referee reports within three weeks from submission. Authors may be requested to submit a revised version of the paper with a reply to the referees' comments to swps@sussex.ac.uk. The Editors make the final decision on the inclusion of the paper in the series. When submitting, the authors should indicate if the paper has already undergone peer-review (in other series, journals, or books), in which case the Editors may decide to skip the review process. Once the paper is included in the SWPS, the authors maintain the copyright.

Websites

UoS: **www.sussex.ac.uk/spru/research/swps**
SSRN: **www.ssrn.com/link/SPRU-RES.html**
IDEAS: **ideas.repec.org/s/sru/ssewps.html**

On the Basis of Brain: Neural–Network–Inspired Changes in General Purpose Chips

Ekaterina Prytkova[1] and Simone Vannuccini[2]

[1]Friedrich Schiller University Jena, Department of Economics and Business Administration
[2]Science Policy Research Unit, University of Sussex Business School, University of Sussex
[1]e.prytkova@uni-jena.de
[2]S.Vannuccini@sussex.ac.uk

February 4, 2020

Abstract

In this paper, we disentangle the changes that the rise of Artificial Intelligence Technologies (AITs) is inducing in the semiconductor industry. The prevailing von Neumann architecture at the core of the established "intensive" technological trajectory of chip production is currently challenged by the rising difficulty to improve product performance over a growing set of computation tasks. In particular, the challenge is exacerbated by the increasing success of Artificial Neural Networks (ANNs) in application to a set of tasks barely tractable for classical programs. The inefficiency of the von Neumann architecture in the execution of ANN-based solutions opens room for competition and pushes for an adequate response from hardware producers in the form of exploration of new chip architectures and designs. Based on an historical overview of the industry and on collected data, we identify three characteristics of a chip — (i) computing power, (ii) heterogeneity of computation, and (iii) energy efficiency — as focal points of demand interest and simultaneously as directions of product improvement for the semiconductor industry players and consolidate them into a techno–economic trilemma. Pooling together the trilemma and an analysis of the economic forces at work, we construct a simple model formalising the mechanism of demand distribution in the semiconductor industry, stressing in particular the role of its supporting services, the software domain. We conclude deriving two possible scenarios for chip evolution: (i) the emergence of a new dominant design in the form of a "platform chip" comprising heterogeneous cores; (ii) the fragmentation of the semiconductor industry into submarkets with dedicated chips. The convergence toward one of the proposed scenarios is conditional on (i) technological progress along the trilemma's edges, (ii) advances in the software domain and its compatibility with hardware, (iii) the amount of tasks successfully addressed by this software, (iv) market structure and dynamics.

Keywords: neural network; Artificial Intelligence, technological trajectory; semiconductor industry; hardware; software

JEL Classification: L63; O31; O33

We would like to express our most sincere gratitude to Prof Edward Steinmueller who provided not only valuable insight and expertise but also relentless support that greatly improved the manuscript. We are also grateful to the anonymous reviewers from the Scientific Committee of EMAEE 2019 Conference and SPRU Working Papers Series for their positive evaluation and generous comments.

1 Introduction

A novel dynamics is unfolding in the semiconductor industry. Producers are attempting to develop a new type of chip to satisfy the needs of downstream markets exacerbated by the rise of Artificial Intelligence Technologies (AITs). Initiated in the 1950s with the Dartmouth Summer Research Project on Artificial Intelligence (McCarthy et al., 1955), the field of Artificial Intelligence (AI) had grown since then, with recent success brought by the so–called connectionist approach that has, at its core, Artificial Neural Networks (ANNs) capable of (statistical) learning. Over the last decade ANNs have proved to be an increasingly effective problem–solving tool to address computation tasks such as natural language processing, speech generation and situated robotics, previously troublesome for the classical computation approach. In terms of information processing, ANNs represent a very different way of algorithm organisation, and, due to them being structurally so different from other approaches, their diffusion triggered a rethinking of the ways in which computation can be organised and emulated in physical (hardware) devices. Such "shock" currently hitting the semiconductor industry creates challenges and windows of opportunities for its players. In the process of dealing with these new challenges, several techno–economic factors and mechanisms that shape the industry's incentives will steer the next phase in the semiconductor industry life cycle toward one of the potential scenarios of development we will outline.

In this paper, we provide both a technological and economic perspective on the path of development taken by the semiconductor industry in the past in order to set up the context in which AITs currently trigger changes. We put together theoretical arguments and empirical evidence about the influential forces and factors in chip production to outline two possible scenarios for the evolution of chips.

To expose the discrepancy currently forming between capabilities of chips and their required performance, we start with documenting existing approaches to problem solving that use different ways of organising computations in order to obtain a result. Then we proceed with an overview of physical implementations of different models of computation into a device to highlight the fundamental connection existing between an approach to computation and an executing computing device. There we explain the dominance of the so–called von Neumann sequential architecture with the fact that it replicates in physical devices the prevalent model of computation rooted into formal logic and rule–based programming. This architecture remained at the core of what we will label the "intensive" technological trajectory followed by the semiconductor industry for decades until recently the success of modern AITs organised in a different algorithmic manner apparently is bringing this trajectory to a halt. We provide an explanation of the distinct nature of ANNs and the radical value added that the combination of ANNs and Deep Learning (DL) brings to the possibilities for the solution of some computation tasks. From the contextualisation of novel computation approaches, we derive the implications for the direction of innovation in hardware architectures and design, and the tensions that this introduces to the construction of chips. We explore potential changes in hardware architecture and design to make chips capable of efficient execution of a growing variety of algorithms; we support our claims with evidence from the industry and patents statistics.

Continuing to form an understanding of the semiconductor industry, we put together the

influential factors and forces at work that affect the technological trajectory chosen by the industry. Given the current crossroad situation begotten by modern AITs, the interaction of these factors and forces will define the future of chipmaking. We summarise the characteristics of a chip, which are at the same time the focal points of demand interest and technological features subject to improvement for chip producers, with a trilemma. The strategy to address the trilemma is influenced by the mentioned economic forces. The interaction between technological factors and economic forces at work in the industry will define which scenario unfolds. To demonstrate how demand distributes between alternative hardware systems (chips), we develop a stylised model that incorporates the trilemma and highlights the fundamental connection between a computing device (chip) and the computation approach embodied in a program. In other words, the model sheds light on the importance of the software industry as a source of indirect network externalities that, in turn, affect the distribution of users among competing hardware systems. Finally, building on the analysis conducted in the previous parts, we conclude with two possible scenarios for the evolution of chips and provide some arguments in support of each of them.

Our study contributes to the nascent literature on the economics of AI (Goldfarb et al., 2019), and it is unique because it focuses on the technological platform (hardware) that works in a closed cycle with AITs: at the same time hardware enables and is affected by AITs. The paper is also a study on the nature of technological trajectories (Dosi, 1982) and the forces that support or contest them.

The paper proceeds as follows. Section 2 considers computation as a problem solving tool and introduces the notion of model of computation, a bridging link between software and hardware domains. Section 3 provides insights on the implementation of models of computation into physical devices (chip) and explains the technological tensions brought by the recent rise of modern AITs; it proceeds with a description of dominant and novel alternative chip architectures and prospective directions of their improvement with chip design. Section 4 outlines the techno–economic factors and forces that characterise the semiconductor industry. There we derive a trilemma capturing in a general manner the trade–offs of chip production, and place it on the crossroad of demand and supply in a stylised model that explains the division of a chip market among alternative hardware systems. Lastly, taking stock of our analysis, we develop two scenarios for the evolution of chips and the future of the semiconductor industry in general. Section 5 draws some conclusions.

2 Computation Approaches to Problem Solving

2.1 Computation as a Problem Solving Tool

People solve a continuum of tasks with the help of computers. In less than a century, computers have gotten firmly entwined with our lives, and computing became an ubiquitous activity. Basically, what we do with the help of computers is problem–solving activity through the formalisation of a task and its structured execution (what is known as an algorithm). A plan is a good example of a problem–solving method. The achievement of a given goal is possible due to the stepwise achievement of subgoals by undertaking actions chosen from a pool of available

actions. A simple mathematical formula like the following is a plan:

$$2^2 + 2 \times 2 = 8$$

Here the goal is to get the final result of calculations, the subgoals are the intermediate components before the summation, the pool of possible actions are the mathematical operations of sum, multiplication and exponentiation, and arithmetic rules set an algorithm of actions. The approach followed to execute the plan can be either a sequential calculation of sum components, or parallel calculation. This formula already represents a particular plan, expressed by the combination of given mathematical operations and the operand 2. The problem to be solved here can be expressed as "get 8 using only 2s by means of sum, multiplication and exponentiation". It is clear that there are other ways of getting the same result. In the case of this toy problem the solutions $(2 \times 2 \times 2)$, $(2 + 2 + 2 + 2)$ provide the same result in respectively two and three steps. These are alternative plans. Note that in the goal setting of this example it is not specified whether or not we are interested in the smallest amount of steps; it is here where the idea of calculation efficiency steps in. In order to call a calculation "efficient" it is necessary to respect constraints of any kind — for example on time, on available actions or operands. In our case we are constrained by three available operations and only one possible operand ("2") so that all the solutions listed above fit. But which solution is more efficient? In order to answer this question "one must have some criterion for efficiency of calculation. [...] [T]o get a measure of the efficiency of a calculation it is necessary to have on hand a method of measuring the complexity of calculating devices..." (McCarthy et al., 1955, p.2). In other words, the efficiency of a task's solution should be assessed in a way that relates both the computation plan (algorithm) and the device on which the computation occurs (the hardware). In a very abstract setting, this notion highlights the complementarity between the hardware and software domain that is key to understand the impact that AITs can have on chips and that we are going to dissect in the proceeding of the paper.

In sum, the set of feasible actions performed sequentially or in parallel with available means and leading to a defined goal is a plan as a problem–solving technique. Written in code (that is, in a machine–interpretable language) a plan becomes a program where the set of actions and an algorithm for their execution are instructions, while operands are data. In general, any program is a *virtual machine* that is ran on a physical machine — a computer. The formalisation of a task implies two mediating steps to be made from the real to digital (thus, computer–mediated) problem solving. The first challenge is the formalisation of the algorithm itself, or the content of a problem–solving technique. In our example, the algorithm is defined by the formal rules of arithmetic operations with some degrees of freedom so that we can decide which alternative plan to follow. The second challenge is the formalisation of the environment, or the context, for the problem–solving technique. For example, encoding information (see Shannon (1948)) that serves as an input for a virtual machine in order to start a goal–seeking process involves the problem of the representation of information.

Symbolic Approach. The are several competing approaches to problem solving. One of them, called the symbolic approach, is considered as classical and based on formal logic and rules. The symbolic approach pioneered many fields of application of computers. Intrinsically

structured tasks — with explicit logic or algorithm's rules and easily codifiable environment e.g. domains of pure mathematics or games — were targeted first. Two prominent examples from 1950s are the Logic Theorist (Newell & Simon, 1956) and its successor the General Problem Solver by Shaw, Simon and Newell (Newell et al., 1959). Both programs were applied to the highly formal tasks of proving theorems by employing propositional logic. However, soon it became evident that for less deterministic tasks an effort to explicitly pre–program all possible actions (operators), operands, intermediate and the final outcomes leads to the problem of combinatorial explosion, as the search space of possible solutions becomes impractically large.

With the growing complexity of the tasks performed by computing systems, the complexity of programs grew substantially as well; hard–coded "expert systems" like the inference–engine–based CADUCEUS or the evaluation function of the chess algorithm Deep Blue are the high–end examples of rule–based programming. However, facing problems of combinatorial explosion, uncertainty and lack of flexibility — and, hence, the incapability to deal with each contingency and the high maintenance required to support effective functioning —, the symbolic approach displayed some of its limitations. In other words, "[l]ogic requires certainly, and the real world simply doesn't provide it" (Russell, 2019, p.40).

Connectionist Approach. Meanwhile, an alternative approach to problem–solving called connectionism was eclipsed by the success of the symbolic one. Connectionism dates back to the 1950s, with ground work of McCulloch and Pitts on neuron–like structures capable of calculations (McCulloch & Pitts, 1943) and Hebb's theory of cell–assembly formation under repeated particular stimulation (Hebb, 2005, p.19,62); the approach acquired its name from the network topology of the core program it relies on: an Artificial Neural Network (ANN). Together, the principles of emulation of logic functions through network connections and the automated assignment and adjustment of weights of connections made it possible to cope with the complexity of some unstructured tasks where the solution algorithm is either unknown or hindered by combinatorial explosion or/and incomplete information. The connectionist methods evolved even during the "AI winter", and currently modern AITs encompass probabilistic tools (statistical and machine learning (ML)) and can work with unstructured data, in contrast with formal logic and explicitly specified functions in the symbolic approach.

Comparing the Approaches. There are different types of problems for which a single or multiple solutions could exist. The solution algorithm can be more or less efficient in terms of time required to compute an answer and/or memory involved to support the computation. To illustrate this statement we consider two examples, the first one is our numerical example, and the second is a task of object recognition in an image.

In our example the arithmetic rules are well–defined and the correct solution can be obtained quickly and definitively. Employing tools of formal logic to code arithmetic rules illustrates the symbolic approach to solving this problem. Now imagine that the arithmetic rules are unknown and that a program can sum before multiplying. In such case many other answers besides 8 are also possible to obtain through exploration. Given that the goal states "get 8", answers different from 8 will be ruled out. Eventually, a program will "learn" the arithmetic rules from multiple examples like ours and will be able to "predict" answers to other equations without having a pre–coded answer. This method illustrates the connectionist approach to solve the

same problem. From the description one can notice that this approach requires more than one example and more time to get an answer and to get it right. Moreover, every solution of the connectionist approach contains some level of irreducible error. In sum, for a numerical task like the one considered, both approaches can deliver the result, with the symbolic approach being more efficient.

Now consider a problem of object recognition in an image, a vital problem for autonomous vehicles. To classify an object as, for example, a pedestrian one has to identify a sufficient set of features that characterises a pedestrian, codify the features and write an algorithm (rules) that evaluates the correspondence and "decides" upon classification. However, every feature of a pedestrian such as legs, hands, etc., has numerous realisations (different sizes, shapes, angles of observation, etc.), and the set of features is subject to uncertainty. In this case to identify some rules ex ante, e.g. "every object bigger than 2.5 meters is not a pedestrian", would make an heuristic shortcut for an algorithm but wouldn't deliver the complete solution. Thus, using the means of formal logic that requires complete certainty about the essential components of the problem setting, this task turns into an immense resource endowment (in terms of time and memory), if tractable at all. Contrary to an explicit formalisation of the features and rules of classification, a connectionist network filters multiple examples through itself and obtains the structure that resembles an implicit, complex classification algorithm. This approach to the problem of pedestrian recognition involves a reasonable amount of time and memory, and in comparison with the symbolic approach in application to this task is incommensurately more efficient.

These examples help to conclude that in approaching a problem with a particular algorithm one faces different levels of complexity or, in other words, different costs to compute the solution on a device in terms of time and memory. In the case of the numerical example, it would be impractical to approach it with the connectionist algorithm in comparison with the symbolic one, facing unnecessary complexity. However, in the case of object recognition, the symbolic approach faces complexity that requires exponentially growing time the farther an algorithm computes, and this is where the connectionist algorithms can fruitfully step in, eliminating exponential growth of compute time[1].

Taking stock of the discussion so far, a task can be approached, among others, in a symbolic or a connectionist manner. Although the symbolic approach became the prevalent one at the outset of the computer era, since the 2000s the growing digital infrastructure (Greenstein, 2019) and the availability of vast amounts of data paved the way for successful deployment of data–driven solutions based on ML techniques, and thus for a re–emergence of connectionism as a paradigm to be adopted alongside rule–based approaches. As this viable alternative gains traction, so does the exploration of the economic activities and new business models based on or centered around it. One transformation currently in the making is the implementation of new computational techniques into physical processing units. How this dynamics is unfolding is the focus of the next Section.

[1]More precisely, transferring the problem from exponential runtime class to to, for example, polynomial runtime.

2.2 Models of Computation

As we pointed out earlier in the paper, the complexity of a given computing technique should be estimated in connection to the device that performs it. Therefore, it is necessary to consider how the structure of a given physical device has been shaped to optimise the joint performance of the device and the number of virtual machines executed on it. Given the prevalence of the rule–based type among virtual machines at the dawn of the computer era, the conceptual structure of early hardware heavily favoured features of the symbolic approach. Thus, in order to understand how the current spike in usage of connectionism–based AITs can lead to potential changes in the life cycle of the semiconductor industry it is necessary to consider the concept of model of computation, which brings together virtual machines and their physical implementation.

In the theory of computation *a model of computation* is the conceptual framework that describes how a result of an algorithm is computed given the available components and their possible interactions. The dominant model of computation implemented in the vast majority of computing devices is the Turing machine. Turing's automatic machine (a–machine) performs computations by scanning one symbol per unit of time from an infinite tape and applying one of its finite configurations (operations) (Turing, 1937, p.231).

Sequential Model of Computation. This model belongs to the class of *sequential models of computation*, which carries both advantages and shortcomings. On the one hand, sequential execution allows for immense flexibility of manipulations over data, making feasible the performance of complex algorithms on a Turing machine, a property which Turing called *universality*. On the other hand, performing a highly complex algorithm in a sequential manner might lead to an impractically long time of execution. Besides, probably the most important shortcoming stems from limitations of formal systems stated in Gödel's incompleteness theorems, the *Entscheidungsproblem* or "decision problem" and the "halting problem".[2] In a nutshell, these problems combined state that there is no way to reliably define whether a program converges to a solution or not. Thus, a formal solution might exist but the convergence to it can take such an immense amount of time that it is impossible to distinguish from absence of solution. Altogether formulated differently, the solution of a problem must rely on an explicit algorithm. In the same paper where Turing described his abstract machine, he comes to the conclusion that both the *Entscheidungsproblem* and the halting problem are impossible to solve on his machine.[3]

Despite these limitations, the Turing machine model of computation was physically implemented in the so–called *von Neumann architecture*. Due to its universality property, the von Neumann architecture proved to be fit for the execution of vast amounts of virtual machines (algorithms) allowing to address a large set of problems, and, due to positive reinforcement between the device (hardware) and virtual machines (software), this set continued to grow over time. For this reason the von Neumann architecture implemented in a chip's Central Processing Unit (CPU) gained a foothold as the dominant model of computation, with its limitations (e.g.

[2] *Entscheidungsproblem* concerns the possibility to prove every step of an algorithm or, in other words, an algorithm can compute a solution of a problem on Turing machine if and only if the all the steps of an algorithm including the final one (the result or solution) can be deduced ("proved") from the set of axioms applying rules of logic. The halting problem says that there is no algorithm that can decide whether to terminate computation because the answer is not feasible (a program goes in an infinite loop) or because it still calculates an answer.

[3] Turing didn't introduce the term "halting problem", he essentially introduced the instantiation of the problem in his paper on the pages 246–248 (Turing, 1937).

the reduction of lead time of sequential processing) circumvented through improvements within the architecture itself and in circuit design.

Following the underlying Turing machine model of computation, programs' logic of organisation under the von Neumann architecture replicates the same computational principles. Traditional programs are sequences of machine instructions with tags indicating which data is needed to perform the respective instruction in a sequence. This sequence has to take into account interdependencies among instructions: if one instruction uses the results of execution of some previous instructions then it has to be placed later in the sequence.

Obviously, the solution of a problem, even when it exists, might be highly impractical to implement because of prohibitively large endowment of resources such as computing power, memory and data required, energy consumption and time. The theory of computation and computability theory address the issues of efficiency of computation (given an algorithm on a chosen model of computation) and of computability of a solution for a particular problem. Thus, some problems can be solved on different models of computation with different efficiency up to the extreme case when one model cannot deliver a solution at all. In applications like signal processing, where data loads are high and lead time is a crucial factor, this efficiency difference plays an important role in the choice of the model of computation and the following circuit design.

In cases when an algorithm is explicit and consists of separable threads, the efficiency of algorithm performance can be improved through hardware (thus excluding improvement through algorithm design) without changing the underlying model of computation by either (i) organisation instruction–level parallelism — that is, the performance of many instructions (quasi) simultaneously —, for example, with pipelining[4], (ii) data–level parallelism — that is, many batches of data processed simultaneously — on a larger number of Processing Units (PUs) implemented on a circuit or (iii) qualitative improvement in a circuit's physical components. In the opposite case, when an algorithm is non–deterministic and cannot be deduced in the form of formal rule, the possibility to write a sequential algorithm for a Turing machine might disappear entirely and, hence, another model of computation can be more appropriate.

Concurrent Model of Computation. *Concurrent models of computation* as alternatives to sequential models of computation are a good candidate for a specific class of tasks. Concurrency as a main principle of such models allows for proceeding with computations along the algorithm by executing several computation threads at different stages without waiting for one another. This implies an inherent modularity, that is, near–decomposability (Simon, 2002), of the algorithm addressing the problem. The modules can be viewed as blocks of an algorithm and can be homogeneous or heterogeneous, or both. The modularity of an algorithm corresponds to the nature of many real–time processes; that is why concurrent models of computation find a use in the context of embedded systems. Applications of embedded systems span from domestic appliances to avionics and AITs, and the choice of the model of computation governing the design of circuits for the chip–using (downstream) markets is of high importance to ensure the

[4] A significant step forward for circuits based on von Neumann architecture was the introduction of pipelining. Pipelining allows for quasi–parallel processing or so–called instruction–level parallelism. Its implementation helped to decrease idle rate and to increase the speed of program processing by employing all functional blocks at every clock cycle.

reliable functioning of embodying technologies.

The problem with the concurrent class of models of computation is that it doesn't have a universal abstraction, a sort of common denominator for this class, unlike the von Neumann architecture for sequential class of models. This implies that circuit design for the concurrent model of computation supports lower universality (heterogeneity of tasks it can execute), and initial attempts to designs such circuits can be tailor–made to a specific family of algorithms.[5] To design and manufacture a circuit entails high costs; hence, to return the investments there should be demand from the downstream markets. Thus, the start of the development process of new circuits depends on the success of specific programs (as problem–solving tools) to gain a critical mass of adopters.

In absence of a universal abstraction that encompasses the whole class of concurrent models, we focus on the *data flow model of computation* because ANNs, the most recent and successful class of programs among modern AITs, are an instantiation of a data flow program[6]. As we concluded in the previous section, there exists a class of problems in which ANNs cope with complexity substantially more efficiently than classical rule–based programs. The increasing availability of data contributes to the growing applicability of probabilistic, data flow methods, such as ANNs. Hence, ANNs have the potential to draw enough attention to the concurrent class of computational models and consequently to trigger and accelerate the development of its physical (hardware) implementations.

In terms of algorithm organisation, ANNs significantly differ from classical programs. An ANN is a multi–layered directed graph. Every layer consists of nodes — instructions represented by some operations over data such as arithmetic functions, e.g. multiply–sum. Connections between nodes in different layers are dependencies between the respective instructions: every possible path in a network is, in a sense, a sequence of instructions. As soon as data is fed to the first layer all first–layer nodes process it in parallel and output the data further to the next layer. Second–layer nodes can start calculating as soon as all required input data arrives. Thus, in terms of execution, a network consist of highly separable computing streams and hence it is inherently parallel to some extent, so that an array of PUs can be fruitfully employed in a concurrent manner. This allows for dynamic scheduling of instructions in contrast to the sequence of traditional programs. Such manner of organising and executing computations constitutes a data flow architecture, where the flow of data defines which instructions to perform; when the data required for the execution of an instruction is ready, this instruction can be initiated without waiting for other independent instructions.

Different from the control flow logic realised in von Neumann architectures, where the data is stable and a sequence of instructions is applied to it (according to the principles of the Turing machine discussed above), in data flow architectures the instructions are stable and the data floats among the instructions. There are several implications for circuit design that can be

[5] "Abstractions that can be used include the event-based model of Java Beans, semaphores based on Dijkstra's P/V systems [29], guarded communication [30], rendezvous, synchronous message passing, active messages [31], asynchronous message passing, streams (as in Kahn process networks[32]), dataflow (commonly used in signal and image processing), synchronous/reactive systems [6], Linda [33], and many others." (Lee, 2002)

[6] In Section 2.1 we refer to a *program* as a *virtual machine*. Boden (2016, p.4) defines virtual machine as "the *information–processing system* that the programmer has in mind when writing a program". Thus, in this paper we use the term *program* in a broad sense to keep the text simple.

derived from this description. Given that multiple instructions are activated by the data flow in a concurrent manner, (i) multiple PUs, and (ii) efficient connectivity between these PUs (see Rent's rule[7]) are required. Together they ensure flexible pathways for data to flow and sufficient processing capacity to perform calculations with low idle rate, timely and without congestions. However, the multiple PUs and their arbitrary order of activation significantly complicate the circuit architecture and at the same time open opportunities for experimentation on how to organise such multiple PUs within a chip's architecture.

Within the concurrent class of models, ANNs rely on their core component which makes them function *autonomously* and effectively — the Deep Learning (DL) technique (LeCun et al., 2015). In a broad sense, every time the obtained answer is correct the ANN retains the structure of its network connections; in case of incorrect answer the ANN adapts. Essentially, DL is realised through weakening or strengthening the connections between nodes of the network propagating the credit for an error backwardly along connections, from the last (output) layer to the first. In principle, connections inside an ANN can be determined by a programmer, and like that, without automated weights–assignment, the ANN as a problem–solving tool would provide only the advantage of higher processing speed due to parallelism. The Pandemonium program of Selfridge (Selfridge, 1958) illustrates this case. To show the power of DL and the key advantage that it brings in addressing problems, we return to our examples from Section 2.1 and consider their execution in a data flow manner only, regardless of the efficiency of this approach in application to these tasks. In the example of the arithmetic task, to predefine manually all connections for a network would be cumbersome but probably feasible. However, to do the same work for the object recognition task would be beyond any feasible effort. Thus, DL makes it possible to fine–tune the structure of the ANN so that it outputs a result with an acceptable level of accuracy, basically grasping patterns hidden in data. There is a whole spectrum of tasks, like the object recognition example, where an explicit algorithm to deliver the solution is either underivable due to complexity or possibly non–existent (for instance, this might apply to task of speech generation). In other words, the algorithm to perform a task is non–deterministic and hence implicit. For traditional programs, the algorithm is a sequence of instructions while for ANNs it is a network's structure of connections. In this case, the ANN plays the role of template for an algorithm and DL is the tool to establish the ANN's structure.

NB: From the discussion so far, we can derive an important conclusion: the value of ANNs resides mainly not on parallelism and its consequent increase in processing speed but on ANNs' ability to create an implicit algorithm to solve tasks for which the explicit approach of deterministic programming is helpless. Parallelism is inherent to ANNs given their network nature as a program, while their Bayesian, inductive essence in achieving the solution is what truly makes ANNs valuable. In a deterministic world, for example, two potential alternative steps in an algorithm can contradict each other; in a probabilistic world of ANNs the same two steps can co–exist, being activated with some probability estimated from previously observed cases (the so–called training phase). As stated above, for ANNs an algorithm is a network structure, hence *parallelism is necessarily present in all ANNs solutions although rather as a requirement*

[7] "Rent's rule is empirical power law introduced in an effort to describe and optimise the wiring complexity of computer logic graphs". (Cuesta et al., 2017)

for obtaining the result than as an advantage in processing speed.

Implications for Industry. Before DL made it possible for ANNs to achieve human–level performance in a set of tasks (for example see Eckersley et al. (2017)), the semiconductor industry had been set on a well–defined technological trajectory with the von Neumann architecture at the core. This was characterised by capacity races (Steinmueller, 1992), non–ergodicity of technological competition, a settled product variety and converged to a stable set of big players. With the boost of development of AITs based on data–driven techniques over the last several years, this technological trajectory is shaken by the need to design chips that would effectively perform new algorithms. As the set of downstream markets for chip producers is already large and continues to grow, this creates the conditions for a so–called dual–inducement mechanism[8] of technological development (Bresnahan & Trajtenberg, 1995) to occur.

Indeed, the recent evolution and extension of the computation paradigm from being mainly rule–based to data–driven poses new challenges for hardware developers. The sequential model of computation appeared to be incapable to efficiently address some problems where the concurrent computational model achieved a certain level of success. The tasks under consideration lie largely in the domain of AITs which is currently gaining momentum. Large digital companies increasingly acquire firms and found departments dedicated to development of AITs. According to the WIPO Technology Trends 2019 report on AI (WIPO, 2019), an evident acceleration of acquisition activity in the AI sector starts from 2012, with 53% of acquisitions (230 out of 434) taking place since 2016. Top–10 acquiring companies are big tech multinationals (in descending order): Alphabet, Apple, Microsoft, Verizon, Amazon, Cisco, Salesforce, Facebook, IBM, Intel, with only Alphabet accounting for 4% of all acquisitions; this statistics underestimates the full scale of acquisitions because it doesn't include non–English speaking countries.

Two remarks can be made at this point. First, as we pointed out before, given that the concurrent class of computation models doesn't have a universal abstraction and as ANNs are currently the most prominent success among other AITs, ANNs exert a dominant pull in the design of circuits, that must be specialised for them. Second, despite the undeniable success of ANNs, truly intelligent machines will require more than ANNs can provide. Quoting Marvin Minsky, "[t]he power of intelligence stems from our vast diversity, not from any single, perfect principle". ANNs triggered a wave of attention to alternative ways of computation and a reaction from hardware producers which cannot be rewound, but the evolution of the field of AI doesn't stop (and most likely will not stop) with ANNs and further solutions will arrive. Problem–solving activity based on computations, and in particular AI, has a long record of being based on and fruitfully employing formal systems; only relatively recently the toolkit of AITs was enriched by ML techniques. It seems plausible, therefore, and many AI researchers share this opinion, that truly intelligent machines will take the best from both symbolic and connectionist approaches. Given that, the response of the semiconductor industry in terms of strategic production decisions must be complex and long–term oriented.

In the next section we shall discuss how some of the existing chip architectures were devel-

[8]With dual inducement it is meant a dynamic (positive or negative) feedback in the coordination game of technological improvement between sectors using the core technology (which, in this case, is the chip) and the sector producing it.

oped and, following the success of ANNs as a prototype of the concurrent class of computational models, which new architectures are appearing. We identify the forces at work, both of technological and economic nature, that have been shaping the semiconductor industry over the last two decades. Eventually, these forces will play a key role in steering the direction of transformation of the industry, triggered by modern AITs.

3 Computation Approaches Shaping Hardware

As discussed in the previous section, the structure of a circuit is fundamentally linked to the logic of computation embodied in the chosen model of computation. If the diffusion of ANNs demands chips with multiple PUs to allow for parallel computation, producers might start exploring the product space along this direction. However, physical implementations face constraints of technological and economic kind.

Undoubtedly, a hypothetical chip with higher (i) energy efficiency, (ii) heterogeneity in terms of types of algorithms it can perform and (iii) computing power could achieve better performance at lower operational costs for the devices embodying it. If programs would not evolve in complexity and increasing diversity, the progress in the semiconductor industry would boil down to miniaturisation of components and replacement of materials that components are made of for better ones. However, not only new "species" of programs had appeared, but also old ones scaled up in size and workloads raising memory and speed requirements as well. Thus, the challenges for the semiconductor industry are posed by the joint pressure of both computation approaches, though with different weights. Increasing volumes, diversity and availability of data coupled with novel ML techniques have created opportunities for new economic activities and business models and augmented some of the existing ones (Agrawal et al., 2019). Among those affected, the semiconductor industry is facing transformation and now responding with exploration of product variety.

In fact, the process of chips transformation has already begun, with early efforts to implement changes in hardware systems accommodating ANNs starting around the late 2000s. Changes can be implemented at the level or architecture and at the level of design (which include new materials, principles and techniques underlying the functioning of chips). We review these in turn.

3.1 Physical Implementations of New Architectures

Before ANNs and DL, parallelism was mainly considered as a way to speed up a program's execution, while with their widespread use it became a *prerequisite* for running certain programs. If the same speedup could be achieved by increasing the speed of sequential processing, then this would require less energy and the system would be easier to manage. Precisely this dynamics has been unfolding in the industry, constituting the well–known Moore's Law. Meanwhile, positive reinforcement between computing devices and programs fueled the intensive trajectory (the one based on the continuous improvement of one known physical implementation of the established computing architecture) over the decades, cementing the prevalence of sequential computing. The pervasive adoption of Internet and mobile communication technologies (Greenstein, 2019)

led to the ubiquity of computing (Weiser, 1999) that scaled up the demand for chips to the global level. Galloping rates of development and the adoption of digital technologies apparently left little time for chip variety development and assessment. Instead, the context pushed the semiconductor industry into serving the needs of growing demand with the von Neumann architecture as a working horse at hand. Thus, the intensive path of technological development was chosen over the extensive one. An extensive path of technological development would have required the exploration of alternative computation architectures and their related physical implementation, a strategy that was judged not viable enough because of the possibility to continue stretching the intensive trajectory and the pressure of markets to obtain a satisfying speed–up with more energy efficiency.

Vector Processors. The idea of realising parallelism in computation in order to increase computing power was, however, already around since the 1970s. For example, products of Cray Research exploited so–called *vector processors*[9]. These processors were capable of operating with a vector as unit operand of an instruction instead of a scalar. Today's Graphics Processing Units (GPUs) embody the same principle. Until mid–2000s vector processors haven't been widespread elsewhere except in supercomputers performing complex computations with large arrays of data, and later in 1990s, with the rise of computer games for the purpose of graphical rendering. GPUs consists of hundreds and even thousands of cores, however less complex and independent than CPU cores. Clearly, the coordinated work of a larger number of cores demands a higher energy consumption and the reorganisation of the computation process according to a specific programming logic; it has to deliver a low idle rate, otherwise the usage of so many cores is not justified (see Amdahl law[10]). Technically, vector processors are still Turing machines replicating the same principle for each of its numerous cores. Thus, their distinctive feature remains only parallelism. GPUs are capable of parallelism on the data level, where one type of computation is performed over batches of data in parallel. This partially corresponds to the concurrency principle mentioned in Section 2.2.

The conventional use of GPUs was as a discrete device on a motherboard for graphical rendering in computer games; however, with the rise of ANNs chipmakers started the integration process of GPUs into a system on ICs in order to exploit GPUs as a new functional module for a broader set of calculations. The set of functional modules that include CPU, co–processor(s) like GPU, memory system, input and output units placed on a single substrate constitutes a so–called System–on–a–Chip (SoC). The integration of a GPU into SoC opened up a potential to effectively include GPUs in computing process for more general calculations, not just operating with graphical data. Using GPUs for general calculations was dubbed General–Purpose Computing on Graphics Processing Units (GPGPU). Bundled together in a SoC, the CPU performs orchestrating work while the GPU executes massively parallel calculations. The main principle

[9] In vector processors, an instruction can fetch a vector and then assign each vector component to PUs to execute calculations in parallel. As an example, the multiplication of two vectors of length 20 in a scalar processor occupies roughly twenty instructions to be executed in a pipelined manner while in vector processor the same multiplication executed in parallel manner takes one instruction.

[10] Amdahl's law describes potential speedup in performance as a function of number of PUs involved (Amdahl, 1967). The exploitation of larger number of PUs represents an "enhanced" or "faster" mode of execution. Thus, said differently,"Amdahl's Law states that the performance improvement to be gained from using some faster mode of execution is limited by the fraction of time the faster mode can be used"(Hennessy & Patterson, 2011, p.39).

of CPU–GPU co–working can be generically described as follows: the CPU dispatches an instruction and points at the related data to the GPU, and the GPU distributes the data across its cores in order to then perform calculations over the data in every core at the same time. The way of computing involving processors with different architectures is called *heterogeneous computing.*

Competing producers developed their own frameworks[11] allowing for heterogeneous computing. Examples are Nvidia's Compute Unified Device Architecture (CUDA) and Fusion System Architecture (FSA), started by AMD, that later transformed in Heterogeneous System Architecture (HSA) by the HSA Foundation, a consortium of companies including AMD, ARM, Qualcomm, Samsung, etc. In 2007, when Nvidia launched CUDA for GPGPU, it was not received with much enthusiasm, certainly not enough to immediately induce a shift from conventional CPU programming to GPU programming. However, Raina et al. (2009) demonstrated the potential of GPU exploitation for ANNs' applications. They showed a method of involving GPU hardware into the training of ANN which surpasses CPU performance in terms of time by a factor from 12 to 72 depending on the ANN's complexity. The joint success of ANNs and GPUs arrived in 2012 from the field of image recognition through the ImageNet Large Scale Visual Recognition Competition. The ANN AlexNet reached on average a 15.3% error rate in classifying 1.2 million images into 1000 categories (Krizhevsky et al., 2012). Since then, ANNs achieved substantial progress, surpassing human level performance in many fields (see Eckersley et al. (2017)). One of the recent examples is the application of ANNs to lung cancer detection improving human expert–radiologists' predictions (Ardila et al., 2019). Another example is the robot BLUE developed by UC Berkeley that exhibits impressive dexterity in dealing with irregular objects using Recurrent Neural Network (RNN) with Deep Reinforcement Learning (DRL). The use of diverse learning techniques with ANNs resulted in an impressive level of autonomy in the performance of tasks to systems where ANNs are deployed. This advantage unlocked new opportunities for a further extension of the boundaries of automation and triggered changes in chips architecture. In short, the extensive path of chips technological development became worth exploring.

Comparing CPUs and GPUs, GPUs consumes more energy but computes faster. Therefore, the same or incrementally larger amount of energy *in total* is required for the training of an ANN which happens within several hours instead of the weeks of training necessary on a CPU.[12] This difference originates from the organisation of computation inside each of the processors.[13]

Array Processors. With the advent of ANNs, GPUs found a new important market; the

[11] The notion "framework" refers to a complex of hardware, programming language and instruction set library.

[12] "The largest model [...] has 45 million parameters, and our GPU method can update these parameters using a million examples in about 29 minutes. In comparison, our multicore CPU takes more than a day per million examples." (Raina et al., 2009, p.5)

[13] A CPU is a *scalar* processor performing one instruction per clock cycle with scalars. A GPU is a *vector* processor with thousands of simpler cores performing the same instruction over each element of a vector in parallel per clock cycle. However, GPU's as well as CPU's cores have to call the data and record results of calculations over the data into memory after every calculation. Such communication with memory implies both advantages and limitations. From Turing's universality perspective on the one hand, this allows having heterogeneous instructions (divide, multiply, AND, OR, etc.) in a sequence. This makes both CPU and GPU capable of general purpose computing although GPU significantly less than CPU. On the other hand, communication with memory after every calculation creates the so–called von Neumann bottleneck, in the setting when memory is represented as a separate block and frequently accessed via bus, limiting processing speed and significantly contributing to energy consumption of a chip.

product was available at the time of ANNs' development and contributed to the breakthrough itself. It didn't take long before the engineering went further. The first serious challenge for Nvidia's GPU came from Google's Tensor Processing Unit (TPU) in 2016. Google's TPU is a *matrix* (or *array*) processor which removes the von Neumann bottleneck from its cores by creating a systolic array of Data Processing Units (DPUs). A systolic array of DPUs represents hard–wired network of homogeneous calculating units — meaning that every DPU implements the same set of operations. Once data is uploaded from the memory it travels among DPUs and is processed upon arrival within a DPU without being recorded intermediately into memory. Thus, the TPU emulates the data flow architecture[14] introduced in Section 2.2. Given that the type of instructions in ANNs is regular, being largely multiply–add operations, DPUs in a matrix processor like TPU perform exactly multiplication and addition. This makes TPU faster in processing than CPU and GPU, performing hundreds of thousands of operations per clock cycle but allows the execution of only regular instructions such as in ANNs (Jouppi et al., 2017).

TPU is more energy efficient and faster in processing due to its data flow architecture but it has the shortcoming of being less flexible or generic in computation. Furthermore, data–level parallelism realised in both GPU and TPU requires the representation of information in regular form of vector, matrix or array in order to effectively run programs (or their parts) on this hardware. This brings us back to issue of representation of problem content and context discussed in Section 2.1. Google improves its TPUs every year, issuing a new generation twice as powerful as the previous one. However, Google has also refrained from the commercial sale of TPUs, using them only in internal services and providing access to customers through the cloud. In contrast, in early 2019, Intel unveiled Intel Nervana Neural Network Processor (NNP) containing both CPU and tensor cores. Nervana NNP is a modified 10^{th} generation Intel Core processor (CPU) with, among other changes, replacement of GPU cores by tensor cores. Thus, Nervana NNP is becoming the first commercially available array processor competing with Google's TPU. A joint project of MIT and Amazon, the chip Eyeriss 2.0 was unveiled in May 2019. It also belongs to the class of array processors with improved connections between spatial array nodes (Chen et al., 2019). Other products announced in 2018 such as Ali–NPU by Alibaba and Inferentia by Amazon will become new in–house chips for the companies' cloud services powered by ANNs.

Neuromorphic Processors. Lastly, a more radical direction of development are neuromorphic chips. This novel architecture mimics the impulse–based synaptic activity between neurons in the brain. Transistors wired together emulate a network of neurons with electrical synaptic connections. Information is encoded as analog signal and flows across an array of simulated neurons in a form of electrical pulses. In effect, the neuromorphic architecture is an analog version of data flow architecture of TPU or NNP. Examples of neuromorphic chips are TrueNorth, a joint venture of IBM and DARPA under SyNAPSE program (Merolla et al., 2014), the experimental neuromorphic chip Loihi from Intel (Davies et al., 2018) and the hybrid (CPU and network of neurons) chip Akida from BrainChip. The distinctive feature of such chips is extreme energy efficiency, which keeps neuromorphic architecture as a prospective competing alternative.

[14]Data flow architecture is inherently parallel but not all parallel systems belong to the class of data flow machines (Veen, 1986).

At the moment, the baseline of embedded systems still relies on a general purpose chip like CPU working in tandem with a specialised one; specialised chips alone do not deliver end–to–end solutions due to the absence of commonly supported and well–developed frameworks. CUDA for GPUs or TensorFlow for TPUs are examples of such frameworks, but they are incomparably smaller than the immense and versatile framework created over the decades for CPUs. The current turbulence and the multiple novel chips introduced in the market may be a temporary phase of fierce competition among tech giants followed by product selection and convergence on a new dominant design. If a successful abstraction for the concurrent class of computation models will be found, this will increase even more the chances of the related chips to gain a foothold in the market. In general, the survival and degree of integration of concurrent models such as data flow into a hypothetical platform chip depends on how many tasks these models can effectively perform which is, in turn, function of how extensive their supporting (software) services/frameworks will be. Thus, the convergence to a stable product configuration is anchored to the success in the software domain, in pioneering new applications (acquiring new tasks) and partially reformulating existing ones ("conquering" existing tasks).

3.2 New Materials, Principles and Techniques

Apart from changes in the physical implementation of computing architecture, chip producers can explore the product space by innovating in chips *design*. This creates the possibility to pursue different paths of chip development on another level (e.g. the very semiconductor material) that do not obligatory touch upon a specific architecture. However, some changes on design level might entail serious implications for the way in which a chip functions. The several alternatives at hand are currently under active discussion in the industry.

Any chip is a substrate that performs operations (executes instructions of a program) over encoded information. The majority of today's chips are *electronic* integrated circuits and work with *digital* signals. Digital signals are made by varying voltage within defined tolerance boundaries; these boundaries define different information states. For example, a binary signal has only two information states: "0" and "1"; hence a voltage below a specific threshold is treated as "0" and above as "1". Encoding information in binary code allows to emulate information in electricity and therefore to introduce it to the hardware for further manipulation.

Transistors in a processor play a key role because their function is to manipulate the flow of electrical current and in doing so to represent information such as data (operands) or logic functions (operations). In a sense, the transistor is the smallest processing unit in a chip. Transistors are electric gates that become open if voltage above a threshold is supplied indicating "1" and remain closed if voltage below a threshold is supplied indicating "0". The possibility to have one of two states "closed" and "open" and to alternate between them is given to a transistor through the properties of a semiconductor material like pure silicon. The transistor has three performance characteristics: (i) conducting speed (frequency), (ii) energy required and (iii) power dissipation. Thus, one of the strategy of chip producers to give their products higher performance while remaining within the energy envelope (i.e. increasing energy efficiency) is to introduce changes in chip's design, for instance through the miniaturisation of chip's components implementing a larger number of transistors and adopting new materials for components that

exhibit enhanced useful properties.

Moore's Law[15] illustrates the effort of chip producers to increase the number of transistors through their miniaturisation. That gives the chip better electronic functionality and, hence, higher processing potential. The number of transistors is increased by reducing their size, which lowers the electrical requirements of a single transistor. However, the increase in the total number of transistors might overwhelm the reduction in electrical requirement per each transistor; therefore, improvements in materials, for example a change in type of semiconductor used to make the transistor, are also required.[16]

Wafer Size. Apart from packing larger number of smaller transistors on a fixed–size chip, there are other ways to increase the number of transistors per chip. One option is to increase wafer size, which could add extra degrees of freedom for chip designers. A larger wafer could be cut into larger dice, which due to the increased area of each die provides additional space for more transistors.[17] Another option is to keep the wafer of the same size but fabricate each chip of bigger size by merging many dice together (each die in such multi–die chip is called a chiplet). The first option faces specific problems: (i) increasing size of a chip and decreasing size of transistors allows for creating potentially powerful chip with however increasing defect rate while printing and decreasing yield at work; (ii) a firm would have to build a new generation of photolithographic equipment with a larger area of printing.[18] Eventually, both ways lead to bigger chip size which entails in either case (i) increasing energy requirements and growing problem of cooling; (ii) dramatically surging costs of having fewer or only one chip per wafer; (iii) compatibility issues (the standardisation of parts) with the device embedding it. Indeed, the increase of Intel's wafer size from 300 mm to 450 mm expected in 2012 did not take place; so far 450 mm wafers haven't been introduced yet. Some examples of increasing chip size exist though. In August 2019, the startup Cerebras released a giant chip exceeding the size of largest GPU by a factor of 56 with 1.2 Trillion transistors and 400,000 AI–optimised cores undertaking the second option we outlined. Cerebras used conventional photolithographic equipment with largest available wafer size of 300mm, and, instead of cutting it, kept it solid by wiring the dice together with Swarm communication fabric to make one chip of 215×215mm size compared to 28.5×28.5mm of the largest GPU. Cerebras markets its product as optimised for ANNs, trying to increase the value of their chip by integrating the frameworks TensorFlow and PyTorch in Cerebras' software platform; this goes in line with the discussion on software–bonded value of a chip mentioned in Section 3.1 and discussed more at length in Section 4.3.

Materials. Ultimately, Moore's Law has production and physical limits. Started in 1995, the pace of progress in production technique with the introduction of a new technology node every two years is slowing down and was expected to arrive at its end in 2014 (Flamm, 2018,

[15]The original Moore's Law was formulated for the highest density of elements on a chip achievable at minimum costs. Even higher density was feasible however only at higher costs (Moore, 1965).

[16]In addition, the power dissipation of modern chips requires improvements in cooling system to avoid overheating and temporary or permanent failure.

[17]A wafer has a circular shape hence its size measures, e.g. 300mm, relates to wafer diameter.

[18]In photolithography the template of patterns for each die to be printed on a wafer is called mask or reticle. Given a chosen technology node e.g. 14 nm, with the help of lensing system this template is projected and printed on the wafer scaled down 4–6 times. A circular area on a silicon wafer within which the printing system with the mask can move is a printing area. This constitutes a complex piece of machinery hence if a change of the printing area is introduced, that would entail a change of the whole system.

p.6). New techniques like Tri–Gate were introduced to extend it until 2025. Eventually, the size of a transistor cannot be smaller than the combined sizes of its components. This is why graphene as a new material has the potential to prolong Moore's Law. A semiconductor like silicon has a crystal with 3D structure, while graphene's crystal is only a 1–atom–thick layer (and it has therefore a 2D structure). Planar components of a transistor made of graphene can reach the smallest possible size, but the research on developing this potential is still ongoing (see the discussion in Schwierz (2010) and an example of physical implementation in Liu et al. (2013)).

Chasing for a smaller size of a single transistor, producers made its parts so thin that it started producing current leakages within a transistor; in other words, electrical insulation has been breaking down, which interferes with a reliable identification of signal levels (the 0 and 1 voltage levels) and raises the energy consumption of a chip. To avoid current leakages which distorted switching precision[19] of transistors and caused overheating and higher energy consumption of the whole processor, in 2007 Intel introduced the technology dubbed High-k + Metal Gate (Bohr et al., 2007). New hafnium– and zirconium–based dielectrics were introduced instead of silicon dioxide serving before as an insulator. The technology earned the "High-k" part of its name due to high values of dielectric constant κ as a physical property of new materials. The other part of the technology, "Metal Gate", denotes the replacement of polysilicon (semiconductor) with metals in order to give higher conducting capacitance and to be compatible with High-k materials.

3D Transistor. A milestone breakthrough happened when in 2011 Intel launched its Tri–Gate technology. Before that, transistor had a conductive channel in the form of thin 2D layer (planar); with the emergence of Tri–Gate technology transistor obtained the third dimension by marginally growing in height but shrinking in width (Auth et al., 2012). This opened up a new horizon for further miniaturisation of transistors: from 32 nm in 2009 to 22 nm in 2011 with the 1st Tri–Gate Generation and to 14 nm in 2014 with the 2nd Tri–Gate Generation. In 2019 a 10 nm specification was introduced[20]. Apart from the size reduction, due to 3D conducting channel a better control over the flows of current was achieved and, hence, faster switching. That resulted in almost 40% increase in processing speed. In the meantime, AMD announced the 7 nm technology node to deploy in 2018; however, it delayed the delivery of the product because GlobalFoundries — AMD's supplier of ICs — revised its roadmap putting on hold the development of 7 nm technique and focusing instead on 12–14 nm technique. AMD is now working with TSMC and in May 2019 delivered the 7 nm specification.

Signal Type. A more radical direction of chip design is related to even more fundamental changes. As explained above, today's chips are *electronic* and work with *digital* signal: electricity encodes information in a discrete way (1s and 0s) with the help of transistors switching on and off under different voltage. Electricity as a way of energy transmission is one of the fastest processing options. However, electricity is slowed down by the medium through which it propagates. A potential improvement is to use light to encode and transmit information because photons are

[19] "We needed a gate insulator that was thick enough to keep electrons from tunneling through it and yet permeable enough to let the gate's electric field into the channel so that it could turn on the transistor." (Bohr et al., 2007, p.4)

[20] It's worth noting that Tri–Gate technology hasn't dismantled High–k technology. Contrariwise, they represent sequential and complementary milestone improvements in chip design.

less subject to interaction with the medium through which light propagates (though has its limitations). Thus, *optical signal* and respectively *photonic* integrated circuits may work even faster. The main engineering problem in this domain is to emulate a transistor optically. This means the impossibility to use a digital (discrete) signal — to encode information in binary code — because a transistor as switcher among 0 and 1 states doesn't exist for optical (light) signal. However, a team of researchers from the company Lightelligence demoed its programmable nanophotonic processor which uses analog (continuous) signal bypassing the problem of optical transistors (Shen et al., 2017). In April 2019 Lightelligence delivered the first prototype of their chip improved with respect to the demo version described in the 2017 paper which, however, is tailor–made to ANNs. It surpasses its state–of–the–art electronic brethren by a factor of 100 in ANN applications, while a comparison on a broader set of tasks has not been done yet.

Interconnect. A growing number of elements on a chip requires more efficient ways of interconnecting them. For decades, the main interconnect structure was a bus. Roughly speaking, a bus represents a one direction highway over which data travels between components of a chip connected to the bus. The bus could effectively sustain a small number of cores. If the bandwidth of the bus is lower than the workload generated by too many cores, the results of calculations from some share of cores have to "wait" to be transmitted back to memory, thus increasing idle rate. To solve this problem, innovative techniques in network interconnect structure were used. Every core or cluster is connected to a router, and routers are combined into a network allowing for concurrency of data transmissions. The network topology of connections allows for optimisation of data transfers and hence faster processing and lower energy consumption. Therefore, the partitioning of a circuit into blocks and the selection of a topology for the network to interconnect them are two important issues in chip design. The next innovation in interconnect was the hierarchy of networks (mesh) creating different levels of networks (Borkar & Chien, 2011; Winter et al., 2010). These networks can function at different levels of voltage, leaving some clusters or cores completely inactive during computations in which they need not to be involved. This helps to save a substantial amount of energy while keeping the computing potential of a chip high. For example, the already mentioned MIT–Amazon chip Eyeriss exploits a two–level hierarchical mesh connecting memory with an array of PUs (Chen et al., 2019).

Die Stacking. Lastly, chip producers found new ways beyond miniaturisation to increase the density of elements on a chip. 2.5D and 3D stacking of chiplets allows not only for tighter integration but also to connect many dice manufactured at different technology nodes and with different architectures (e.g. CPU–GPU), creating truly heterogeneous systems. However, there is a difference between these two technologies. 3D stacking actually piles dice one above the other. 2.5D stacking is rather an integration which uses a common wafer with built–in silicon interposer as interconnect medium and Through Silicon Vias (TSVs) as pins to connect dice to this medium; placed on the common wafer closely side by side, dice are connected through the silicon interposer due to TSVs. Shorter inter–die connections increase processing speed and reduce energy consumption in a heterogeneous system. In August 2014, Intel introduced its 2.5D technology dubbed Embedded Multi–die Interconnect Bridge (EMIB) with the core difference that a silicon interposer in form of a square was replaced with small pieces (bridges) of silicon embedded only under the edges of each pair of interconnected dice. This more flexible structure

allows for diverse layouts, facilitating more combinations of heterogeneous cores. For example, EMIB was used in Intel's Quad Core processor combined with AMD's Radeon Vega GPU. Later it was used in Intel's Nervana NNP mentioned in Section 3.1. Chip–on–Wafer–on–Substrate (CoWoS) is a competing proprietary 2.5D technology owned by TSMC.

3.3 Novel Architectures in Patent Statistics

Over the last 5 years many novel chips entered the market to accommodate the growing data workloads and increasing computing power requirements posed by recent advances in AITs. In particular, producers experiment with the fundamental principles of chips' architecture and design to meet the needs of ML and ANNs while trying to keep their products energy efficient, heterogeneous in terms of types of algorithms to be performed and computationally powerful.

The results of such efforts can already be detected in the IPR system. In January 2019, changes in the International Patent Classification (IPC) were introduced, adding the new G06N 20/00 group[21] which includes patents filing on computer systems based on ML as a specific computational model. Related to hardware, the group G06N 3/02 corresponds to computer systems based on neural networks models; this exhibits an exponential growth since around 2010. Simultaneously, the group G06N 5/00 representing knowledge–based computational models (i.e. expert systems mentioned in Section 2.1 and belonging to the rule–based approach to AI) experienced a growth phase in the years from 2000 till 2008 and currently stagnates. Figure 1 represents these dynamics and Figure 4 shows the distribution of applications across countries to provide insights on the geographical distribution of IPR protection related to novel ICs architectures.

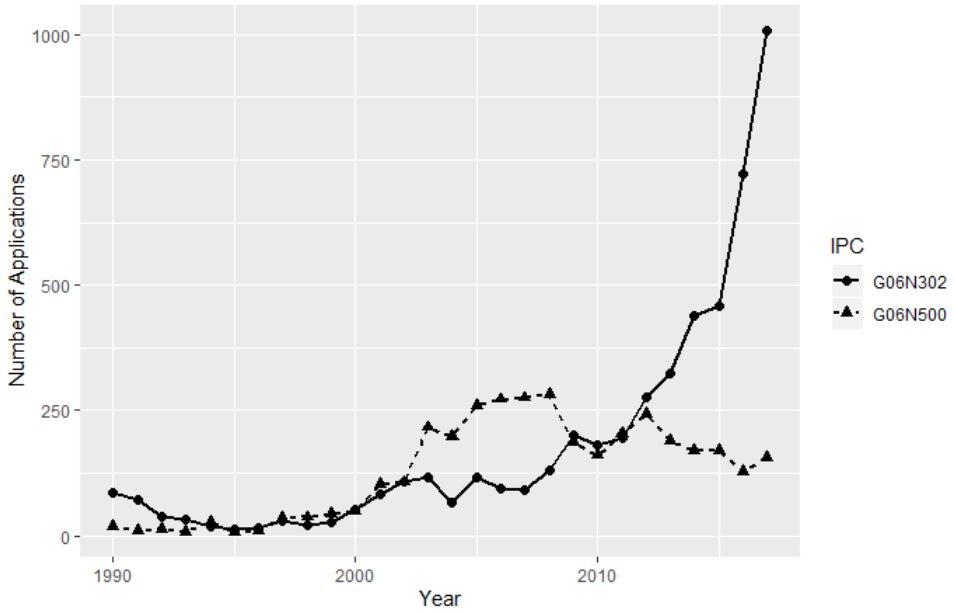


Figure 1: Number of patent applications related to Integrated Circuits (ICs) based on specific computational models over period 1990–2017: G06N 3/02 – ANN models; G06N 5/00 – knowledge–based models
Data: PATSTAT Global 2019 Autumn

[21] According to the information posted on WIPO website, the classification of documents published before January 1, 2019 has not been finished yet, hence we do not include any data for this group in the paper.

In general, the IPC 4–digit group G06N contains information directly relevant to chips and the underlying computational models. However, AITs induce changes on multiple levels of chips' technology that are spread across different subcategories, from basic elements of design like transistors to the overarching architecture and techniques of data representation and processing. Only a share of these changes is reflected in the G06N class. Looking at the broader picture of AITs' impact, beyond chips, WIPO report shows evidences of ML being a dominant AI technique while "other AI techniques such as logic programming (with 99.5% of patent families related to expert systems) and fuzzy logic show a steady filing rate since the late 1980s. A recent, though moderate, increase in patent filings is evident for these two techniques, with between 1,500 and 2,000 priority filings in 2015 and 2016. However, other AI techniques, namely ontology engineering and probabilistic reasoning, represent a very low number of filings in the field (less than 1 percent of all patent families)." (WIPO, 2019, p.41). Thus, the prevalence of ML– and ANNs–relevant inventions over other AI techniques, including formal–logic–based ones, is present in both the specific circuit models of computation (G06N class) and in the overall range of AI–related inventions (e.g. G05B, G06K, G06L, G06T).

4 The Future of Chip Production: Drivers and Scenarios

In previous sections we discussed the changing equilibrium between models of computation pulled by AITs and how this reverberates on the choice set of chip producers for what concerns the architecture and design of their products. In this section we construct an economic framework for the process of chip development.

First, we derive a technological trilemma decomposing the value of a chip for a consumer into its performance characteristics and their interrelations. The trilemma defines technological determinants of demand that can be manipulated by the supply side. This bridges the technological analysis of previous sections with the economic forces shaping together the semiconductor industry.

Second, we outline a set of theoretical building blocks from the economics of technological change that, combined, help us to explain the path taken by the industry, its persistence and the current challenge posed simultaneously by the thinning of available technological opportunities and the increasing pressure exerted by the diffusion of AITs. Interpreting the events of the semiconductor industry through the lenses of economics theory allows us deriving useful insights to understand potential scenarios for the industry's future development.

Third, we proceed further in the formalisation of the dynamics unfolding in the semiconductor industry and present a model of demand distribution driven by the trilemma–based value of a chip. The model stresses and illustrates the role that the software domain plays in guarding the dominance of chips with von Neumann architecture.

Finally, drawing on the analysis of the technological and economic factors and mechanisms, we derive two scenarios for the evolution of the semiconductor industry and point out issues for further discussion.

4.1 The Technological Trilemma

In Section 3 we mentioned three end–characteristics of a chip that are simultaneously technical challenges for producers to address and determinants of consumers' choice. These characteristics are (i) *computing power*, (ii) *energy efficiency*, and (ii) *heterogeneity/flexibility of computations*. In a nutshell, computing power can be viewed as a quantitative measure representing the achievable scale of computation per unit of time, while heterogeneity is a qualitative measure in the sense of how many types of computations or algorithms can be effectively executed using that chip. Thus, heterogeneity heavily relates to Turing's universality.

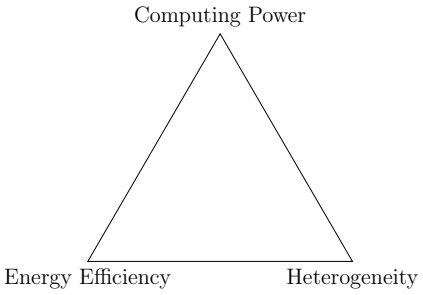
The von Neumann architecture at the core of the majority of chips provided a sufficient level of heterogeneity for many applications over 40 years up until approximately year 2010. During this period, the pursuit of miniaturisation strategy through scaling down the size of transistors and hence doubling their number provided a 40% increase in speed while keeping the energy consumption constant.[22] In other words, it was possible with one move (transistor miniaturisation) to achieve improvements in two out of three characteristics, computing power and energy efficiency, without harming the third one, heterogeneity. However, approaching atomic scale, transistors size reduction cannot continue at the same pace. Producers have now to decouple speed and energy and look for other techniques to improve each of them, possibly not both at once anymore. In turn, the von Neumann architecture was kept at the core while growing in complexity exploiting diverse techniques such as pipelining and out–of–order execution.

Relentlessly pressed by demand's needs, architectures grew in complexity and encompassed multiple functions in one chip, ensuring in this way CPUs dominant position in the industry over such a long period of time. However, the more complex architectures are implemented by involving more elements — e.g. heterogeneous logic cores and on–die memory (cache) — the more energy they require. This sets the heterogeneity aspect in a direct trade off with energy efficiency. Indirectly, and returning to the discussion on algorithms, heterogeneity makes it possible to run sophisticated algorithms, but it is not necessarily associated with high speed of processing them; another emerging tension is therefore that between heterogeneity and computing power. In general, an excellent discussion of chips' technological characteristics with deeper engineering insights is provided in Borkar & Chien (2011). Taking stock of the discussion above, computing power (speed) and energy efficiency were subject to improvement since the dawn of the semiconductor industry, while heterogeneity appeared more recently as a strategic dimension due to the growing diversity and complexity of tasks and, hence, algorithms to which chips are applied. While not a strong constraint earlier on, heterogeneity is increasingly becoming a vital concern for chip producers and its relevance is especially exacerbated by the increasing amount of businesses (startups as well as incumbents) implementing modern AITs (see Sections 4.2 and 4.3 in Perrault et al. (2019)).

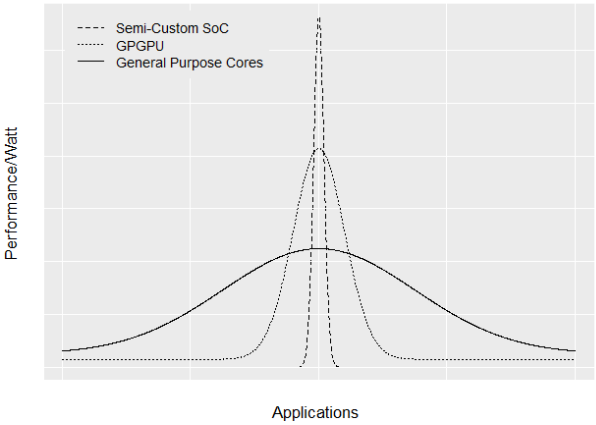
We consolidate the characteristics just discussed into the technological trilemma that chipmakers face when choosing the direction of innovation to increase the value of their products for

[22]Here we refer to Dennard scaling rather than to Moore's law. Moore's law is an empirical regularity relating time and feasible density of elements on a circuit at minimum cost. Dennard scaling is a scaling law based on formal physical principles. It states that (i) reduction of a transistor's dimensions allows shrinking its overall area hence higher density of transistors on a die is feasible; (ii) consequently, the transistor's channel length and interconnections reduce as well, reducing the time of switching (frequency) and transmission of current across the circuit; (iii) this allows lowering the voltage and hence energy consumption (Dennard et al., 1974).

consumers. Computing power, energy efficiency, and heterogeneity represent the set of parameters and intrinsic technical tensions that drive a chip's development. The trilemma works as an identification mechanism for the point of least resistance on which producers allocate efforts in order to expand the technological frontier. In Figure 2 we put side by side two representations of the trilemma. The first one in panel 2a, constructed for this paper, captures the generic idea of pair–wise trade–off between the three constituting characteristics. The second one in panel 2b is reproduced from a recent AMD's keynote address at a symposium on high performance chips held at Stanford University (Su, 2019). AMD's representation additionally places examples of chip types into the same trilemma; applications on the horizontal axis represent heterogeneity, and performance/Watt on the vertical axis combines computing power and energy efficiency. As expected, General Purpose Cores (CPU) have the widest support of application but in terms of performance/Watt falls behind GPGPU and Semi–Custom SoC (highly specialised circuits tailor–made for a particular application, also called ASIC, e.g. TPU). The convergence of different studies and sources on chips' relevant characteristics supports the construction of the trilemma and its implementation in a stylised model in Section 4.3.



(a) The techno–economic trilemma of chip production



(b) AMD's representation of the trilemma

Figure 2: Different representations of the trilemma

The trilemma as a tridimensional performance frontier has both a technological and an economic nature; in fact, the possibility to improve chips on either of the three dimensions depends on technological advances in both hardware and software, the characteristics of demand (shape, preferences, elasticity), and their interaction. We can interpret the trilemma as a tool to understand how producers can reduce the search space for new devices: in order to become a viable product architecture capable of gaining a critical mass of consumers to survive on the market, a chip has to exhibit a Pareto improvement with respect to any of the trilemma's characteristics.

4.2 An Economics of (Chips) Technical Change

In the previous sections, while describing the alternative paths chip producers can follow to steer the development direction of the industry, we briefly touched upon several determinants and strategic "levers", whose viability has been in many case rejuvenated by the diffusion of

modern AITs. A first tool in our hands is the technological trilemma we developed earlier on, that highlights the shape of the technological frontier for innovation in chipmaking.

Now, in order to gain a full understanding of chips evolution, it is necessary to single out in a more holistic way the forces and mechanisms that determined the persistence of the von Neumann architecture over time to assess if and how this persistence can be notched by contemporary pressures. To do that, we make use of some conceptual building blocks derived from the economics of technical change.

Technological Trajectory. The first element we use is the concept of *technological trajectory* (Dosi, 1982). A technological trajectory is, in broad terms, the prevailing line of development followed by a given technology, exemplified by a given characteristic or parameter (e.g. size, strength, resistance, capacity, etc.). Chipmakers have followed the intensive trajectory of elements miniaturisation that allowed obtaining planned improvements in energy efficiency and performance speed at expected costs, and this resulted in Moore's Law. The routine strategy to be followed became that of increasing the density of components in a non–specialised chip that encompasses multiple functions. At the same time, potential alternative architectures might not have been technologically or economically viable.

Competing Technologies. In this context, the second concept entering into play is that of *competing technologies* (Arthur, 1989), that sheds light on the non–ergodic nature of technological competition. The magnitude and certainty of CPUs improvements resulting from pursuing the intensive trajectory compensated the limitations (especially on speed) of the sequential nature of the von Neumann architecture compared to the yet uncertain gains from alternative architectures. Said differently, alternative architectures didn't prove to be superior to the von Neumann architecture in an overwhelming majority of applications apart from very specialised and hidden niches (e.g. high performance computing (HPC)); more applications were yet to be found. By rapidly defining a technological trajectory for improvement with respect to the trilemma's parameters, CPUs prevailed over potential alternatives due to dynamic increasing returns in performance. The change currently taking place in the semiconductor industry originates from the fact that modern AITs demonstrated the useful potential of concurrent models of computation and called for supporting hardware architectures. By gaining traction and applications, modern AITs are questioning the benefits of the prevailing technological trajectory in hardware and pointing at a promising direction for high returns on alternative architectures that started influencing chipmakers' strategies.

Dominant Design. Besides following a predictable technological trajectory, chips based on the von Neumann architecture can also be considered as examples of dominant designs (Suárez & Utterback, 1995). A *dominant design* — our third theoretical building block — is a stable architectural configuration that emerges as a focal end–point for both the supply and demand, given the dynamic reinforcement process of competition and adoption driving the selection of one alternative design among the many existing. A dominant design is hard to break, as it is a form of "inflexibility" (Cantner & Vannuccini, 2017), and it usually imposes high–entry costs for firms that would like to start production but did not contribute to the establishment of the dominant design blueprint. Despite its stability, a dominant design can be broken, either through disruptive innovations "crawling up" the product quality ladder or by the expansion of the set

of tasks that a product is expected to execute. Indeed, a growing set of tasks that turn into more complex and diverse algorithms, including but not limited by algorithms that constitute AITs, exerts an increasing destabilising pressure on the dominant von Neumann architecture as both users and producers (incumbent firms and potential entrants) find increasingly appealing and beneficial the exploration of new architectures. This re–discussion of the dominant design of a chip can potentially lead to the rejuvenation of the industry life cycle in the semiconductor industry.

Architectural Innovation. A fourth useful notion is that of *architectural innovation* (Henderson & Clark, 1990). An architectural innovation is one in which the core function behind a technology remain unvaried while the linkages between its parts change; architectural innovations can represent a subtle challenge for incumbent firms, as they reshuffle established know–how without being as "dangerous" as radical innovations are. The transformation currently taking place in chipmaking can be positioned in–between the notions of architectural and radical innovation. In fact, the core principle of electronic–based computation of digital signals remains intact (except for new, more radical designs outlined at the end of Section 3.2) while the logic of execution and ways of components' connectivity is passing through a period of experimentation. The resulting balance between radical and architectural innovation in chip production can steer the path towards a novel dominant design, envisaging one of the potential scenarios we discuss later in Section 4.4.

Path Dependence. A fifth, and more encompassing concept that knits the threads of the previous building blocks is that of *path dependence* (Cantner & Vannuccini, 2017), where "[a] dynamical process whose evolution is governed by its own history is 'path dependent'." (David, 2007, p.92). Path dependence is moulded by the intensity of technological competition among alternatives and the clarity of technical improvements along technological trajectories. Using the terminology of Page (2006), the cause of path dependence in the case of the semiconductor industry is a combination of positive feedback, or positive/network externalities (the reward of making a choice is a positive function of the amount of users making the same choice) and self–reinforcement mechanisms, according to which a given choice set the stage for other, interconnected choices to be made, which in turn create a sustaining environment for the original action. From the path dependence perspective, one of the forces keeping chip production on a defined and established path has been the relationship with the hardware's "supporting service": software development. Mentioned earlier in the paper, the relationship between computing device and programs (virtual machines) determined the joint performance and fueled a specific technological trajectory.

The "first–mover" advantage of the von Neumann architecture, for all the reasons we discussed in the previous sections, induced the generation of architecture–compatible software; that became the installed base raising the returns for chip producers to stay with this architecture and continue the "intensive" path. As it is discussed in the literature on multi–sided platforms (Belleflamme & Peitz, 2018), the creation of a "network" on a side of the market can solve the so–called chicken–and–egg problem and make a technology or a product that is subject to feedback and reinforcement viable for production. Understanding how software ended up playing an interlocking role in the development of the semiconductor industry is of fundamental

importance: as alternative architectures gain room for experimentation and production given the increasing interest in AITs, the outcome of their technological competition with existing products can be determined by the capability of these novel architectures to launch the same cycle in symbiosis with software and software platforms (recall our mentioning of the CUDA framework) that then spur further positive feedback and self–reinforcing growth.

Production (Dis)economies. The last piece of the puzzle to assess the techno–economic forces shaping the future of chips has to do with the very *production economies* and *dis–economies* encountered by chipmakers that contributed to the persistence of dominance of the von Neumann architecture and its linked intensive technological trajectory. Steinmueller (1992) discusses the choice that semiconductor companies make regarding their production strategies. One option is to intensify production within a firm's set of products in order to exploit economies of scale; the other is to widen the set of products produced, hence to increase variety in order to exploit economies of scope. Chipmakers need to pick one of the options, as economies of scope fueling product variety and economies of scale fueling production expansion are at odds with each other. The semiconductor industry has mostly pursued economies of scale. In fact, chip production is characterised by so–called *capacity races*, namely the setting in an industry pushing for mass production to amortise large costs of equipment capable of little flexibility. The cited example of the unprecedentedly big chip produced by Cerebras fabricated on conventional photolithographic equipment points at the persistent importance of this economic factor.

Capacity races produce two economic effects; the first one concerns the incentives for firms to be a first mover in innovation by preempting competitors in the introduction of new generations of chips (manufactured with improved characteristics historically labeled according to transistors gate length — e.g. 45nm technology node). The rationale behind capacity races is the fact that investment in capacity produces dynamic economies of scale, or economies of learning; hence, by being a first mover, a chipmaker can reap additional payoffs. The second effect is the incentive to push for cost reduction in order to hasten the scaling up of production. This incentive was particularly strong for the semiconductor industry, as each technology node had a short life cycle due to Dennard scaling, the need to accommodate an ever–increasing demand for computing power and not to fall behind in the fierce competition amongst producers for this demand.

In sum, the economic features of chip production, namely high equipment costs and acceleration in products' turnover, formed capacity races, that in turn are characterised by economies of scale and dis–economies of scope and thus called for clear–cut product standardisation. These economic conditions, together with technological factors discussed in Section 4.1, set out a precise track of evolution for the industry over the past 40 years, tying the technological trajectory to the von Neumann architecture and to miniaturisation as its main innovation direction.

The conceptual building blocks outlined above are essential to draw a picture of the potential future of chipmaking. Taking stock, the notions of technological trajectory, competing technologies and path dependence influenced by software–related externalities help to explain both the reasons for the persistence and the challenged position of von Neumann architecture. We highlighted the chance for that trajectory to come to a halt due to the "puzzle" created by the increasing use of AITs and the corresponding hardware architectures. The concept of dominant design is instead instrumental to explain the difficulty in breaking dominance of established and

stable architectures, as they are the backbone of the mature phases of the industry life cycle and represent the blueprint for many downstream markets. At the same time, a dominant design can be challenged when the conditions for its stability are broken by a technological shock. Finally, architectural innovations in the implementation of computation logics on a chip capture the current phase of experimentation with type, interconnection and composition of components of a chip.

We can summarise our analysis as follows: the development of new chips is bounded above (upstream) by fixed equipment costs, and below (downstream) by network externalities in software, which in turn are affected by the demand for computing tasks to be performed. Within these bounds, the specific path taken has been shaped by a co–evolution dynamics. Such dynamics is characterised by competition between alternative technologies and the forces ensuring a path dependent development along a defined technological trajectory. The economic trade–off between exploiting scale within the prevailing technological trajectory or exploring variety of other architectures with the risk of failure and lack of demand has been characterising the semiconductor industry for decades. What makes the current period unique in the life cycle of the semiconductor industry is that modern AITs have the potential to break the established intensive technological trajectory.

The success of modern AITs favours alternative chip architectures and confronts chip producers with the trilemma, whose structural tensions are currently exacerbated as heterogeneity gain relevance amidst increasing complexity of algorithms and tasks. In case the established trajectory is indeed broken, the process of transition and its destination trajectory will be defined through the interaction of the following aspects:

(i) feasible degree of addressing the trilemma: e.g. improvement in one characteristic of the trilemma at a time while keeping the other two intact or the advancement in two characteristics at the cost of the third one, etc.;

(ii) approach: applied — e.g. design of chip's elements, materials, architectures; fundamental — e.g. models of computation, physics principles.

Decisions of chip producers on each of these aspects is a strategic matter; they are guided by intertemporal cost/benefit analysis and by the expected dynamic returns of pursuing the chosen depth (applied or fundamental) and scope (one, two or all three characteristics addressed) of experimentation. On the one hand, capacity races and path dependence, additionally reinforced through the software domain, might still remain strong forces supporting more "old–fashion" improvements, like those that supported the intensive technological trajectory over the last 40 years, which imply lower costs and lower economic risk. On the other hand, there is a clearly detectable trend of "innovation for buyout" among startups that seek for being acquired by large firms (Cabral, 2018); the presence of this tendency among new firms increases the diversity of ideas and the amount of experimentation at a smaller scale. If these are successful or promising, acquisition allows for scaling up. Acquisition statistics from the WIPO report, mentioned in the end of Section 2.2, support the claim that such trend indeed exists. This setting raises the likelihood of bolder and risky (and costly) moves, for example a more fundamental approach to chip development aspiring at addressing all corners of the trilemma.

Once production decisions are taken on the two aspects listed above, they shape the next generation of products, novel or upgraded ones, that appeal to different shares of demand, indicating in turn the products' economic viability. This can create a repeated game with multiple stages in which chip producers adjust their production and innovation strategy according to the response of demand and observing the products of their competitors. The real–world example of such behavior is Intel's Process–Architecture–Optimization strategy that was implemented in 2016, replacing the so–called Tick–Tock strategy. At the first stage, the company focuses on the technology node increasing the density of elements on a chip to increase computing power and lower energy consumption. Then, at the second stage, the company develops a new architecture exploiting a larger number of elements to form blocks and connects them introducing newly added functions. At the third stage, the company optimises the work of the whole chip.

4.3 Model

In this section we introduce a simple model to show how demand distribution is affected by the technical characteristics of the trilemma and the outcome for the industry players. The model rationalises the interaction among several of the techno–economic forces we outlined in our analysis.

We include the software domain into the model for two reasons. First, it allows for an indirect modelling of the heterogeneity characteristic, approximating a chip's technical capability to run a variety of programs. Second, it reproduces the feedback loop between the software domain and the semiconductor industry that fuels path dependence, in line with the argumentation of the supporting services approach. The supporting services approach is also referred as indirect network externalities, where consumers are indifferent to number of users of a product but are interested in the variety of services that this product gives access to. We ground our model on the framework provided in Shy (2011) and modify it for the case of chips by (i) modeling consumers' utility through the trilemma–based value of a chip, (ii) considering partial compatibility (Chou & Shy, 1993) and (iii) deepening the interpretation of some parameters due to industry–specific features. In what follows, first we outline the model and then we comment it, supporting the results with evidences from the marketplace.

Demand Side. We assume that consumers are uniformly distributed over a unit interval and indexed with x. Consumers can be considered as individuals or downstream markets. They buy only one chip each, making a choice between a chip i and a chip j. For illustrative purposes we will also employ the notation with chip C where C stands for "control–flow" and chip D where D stands for "data–flow". The parameter δ is usually interpreted as degree of product differentiation. In the case of chips, the parameter δ measures the disutility from purchasing a chip that doesn't completely match with the type of computation it is bought for by a consumer. For example, if a consumer needs a chip for mainly control–flow–organised computations and only a small share of the consumer's calculations would benefit from a data flow chip, the parameter δ reflects the reduction in utility from buying a non–perfect match to the consumer's needs.

$$U_x = \begin{cases} V^C - \delta x - p^C & \text{if buys control–flow–organised chip} \\ V^D - \delta(1-x) - p^D & \text{if buys data–flow–organised chip} \end{cases} \quad (1)$$

Equation (1) represents the utility of a consumer from buying one of the alternatives, where (V^C, p^C) and (V^D, p^D) are, respectively, the value and the price of control–flow–organised and data–flow–organised chip. The values are described as:

$$V^C = E^C S^C \quad (2.1)$$

$$V^D = kE^C S^D \quad (2.2)$$

The value of a chip relates to the trilemma described in Section 4.1. The rationale behind the E component is as follows: the computing power or performance of a chip is measured in operations per second. The energy efficiency of a chip is basically its energy consumption per unit of time, expressed in Watts (W).[23] Thus, we introduce the combined efficiency measure E obtained by dividing performance (in operations/s) over energy efficiency (in W), merging two characteristics of the trilemma into one. Note that the higher the energy efficiency, the smaller its measure in W. This or similar measures are indeed used in the industry. For example, the recent analysis of Open AI on amount of compute shown by modern AI systems uses FLOPS/W[24] as performance measure which, it is argued, is also correlated with FLOPS/$ (Amodei et al., 2019). Any of these measures fit in the model's logic. The parameter k is a scaling parameter that helps to express one chip's efficiency in terms of the other chip's efficiency, i.e. $E^i = k \times E^j$. For example, if $k = 2$, it means that chip i is twice as good as chip j by either performing twice as many calculations with the same energy consumption or consuming twice less energy to perform the same amount of calculations.

The remaining characteristic of the trilemma, heterogeneity or flexibility of calculations, is more subtle to model. From the discussion in Section 2 we know that some programs can be addressed through either approach (sequential or concurrent, symbolic or connectionist) and hence performed on any type of chip, however with a sheer difference in time and/or energy endowment. Therefore, there is some degree of interchangeability between chip types expressed in terms of software. Note though, that some tasks are chip–specific due to either inherent technical features of a program (e.g. for ANNs parallelism as a requirement to converge to a solution in reasonable time) or because of yet the absence of effort to reprogram these tasks for the other type of chip. In practice, what matters is how many programs or tasks can be performed using a specific chip within a reasonable span of time and energy envelope. Therefore,

[23] More precisely, under energy consumption we mean the amount of energy required to move an electric charge, expressed in joules (J). Thus, energy efficiency can be measured in joules per second which is equal to Watts: $W = \frac{J}{s}$.

[24] FLOPS stands for floating point operations per second.

the heterogeneity of a chip is modeled as follows:

$$S^C = s^C + \rho^C s^D \quad \rho^C \in [0;1] \tag{3.1}$$
$$S^D = s^D + \rho^D s^C \quad \rho^D \in [0;1] \tag{3.2}$$
$$s^C + s^D = 1 \tag{3.3}$$

The total amount of software that can be run on a chip S^i where $i = C, D$ consists of two components: (i) the amount of chip–specific software s^i, (ii) a share of software written for the other chip that can be interchangeably run on both chips $\rho^i s^j$. The total amount of tasks/programs to be performed is normalised to 1.[25] The parameter ρ^i reflects software's *partial* ($\rho^i < 1$) and in most cases *asymmetric* ($\rho^i \neq \rho^j$) interchangeability between chips. In sum, the magnitude of S^i approximates the heterogeneity of computation that chip i provides. One can reasonably assume that even though the control–flow–organised chip cannot handle all software written for the data–flow–organised one, it will still have a relatively higher interchangeability potential compared to the data–flow–organised chip, which implies $\rho^C > \rho^D$.

The difference in values of the two chips can be written down explicitly:

$$\begin{aligned} V^C - V^D &= E^C S^C - kE^C S^D \\ &= E^C(s^C(1 - k\rho^D) + s^D(\rho^C - k)) \end{aligned} \tag{4}$$

To analyse the comparative statics of the value difference shown in equation (4), we simply take partial derivatives with respect to each variable in the expression.

$$\frac{\partial(V^C - V^D)}{\partial k} = -E^C(s^C\rho^D + s^D) = -E^C S^D < 0 \tag{5.1}$$
$$\frac{\partial(V^C - V^D)}{\partial \rho^D} = -E^C s^C k < 0 \tag{5.2}$$
$$\frac{\partial(V^C - V^D)}{\partial \rho^C} = E^C s^D > 0 \tag{5.3}$$

Equation (5.1) shows that the more efficient (higher k) the data–flow–organised chip in terms of combined performance and energy efficiency in comparison to the control–flow–organised chip, the smaller the gap between chip's values, other things equal. An increasing capability of the data–flow–organised chip ρ^D to run software written for the control–flow–organised chip s^C also reduces the gap between chip's values in favour of the data–flow one. Contrariwise, increasing ρ^C leads to growth of the gap in favor of the control–flow chip, other things equal. The two derivatives with respect to the ρ parameters can be interpreted as the attempts of producers to invest in architectural improvements in order to incorporate the functionality of the competing chip and therefore to manipulate the indirect network externalities.

The remaining two variables s^C and s^D can be interchangeably expressed according to (3.3), hence $s^C = 1 - s^D$ and $s^D = 1 - s^C$. Using this substitution and taking partial derivatives, we

[25] We purposefully do not model a law of motion for s^i, as we are interested in understanding the allocation of users across systems given a set of available supporting software and their degree of "multihoming" captured by the ρ parameters.

obtain the following:

$$\frac{\partial(V^C - V^D)}{\partial s^C} = E^C k(1-\rho^D) + E^C(1-\rho^C) > 0 \tag{6.1}$$

$$\frac{\partial(V^C - V^D)}{\partial s^D} = -E^C k(1-\rho^D) - E^C(1-\rho^C) < 0 \tag{6.2}$$

These equations show that with a growing amount of software written for the control–flow chip s^C the gap $(V^C - V^D)$ increases, while the opposite holds for s^D. In general, the technical superiority of a chip i in terms of performance per Watt $E^i = kE^j$, along with more tasks supported on this chip s^i increases its value V^i and hence increases the gap $(V^i - V^j)$. If the software interchangeability parameter of a chip ρ^i would be equal to 1, that would mean that chip i is capable of performing all the tasks that chip j does. In other words, if $\rho^i = 1$, the chip i would support the highest possible heterogeneity of computation. However, precisely the imperfect interchangeability of chips, captured by $\rho^i < 1$, doesn't allow for completely dismantling either of the chips. Lastly, it is worth stressing that here we want to analyse the dynamics of the values gap when varying each of the component. Therefore, while the expression $(V^i - V^j)$ can grow along, for example, s^i, the absolute value of the gap can be any, positive or negative, namely $(V^i - V^j) \gtreqless 0$.

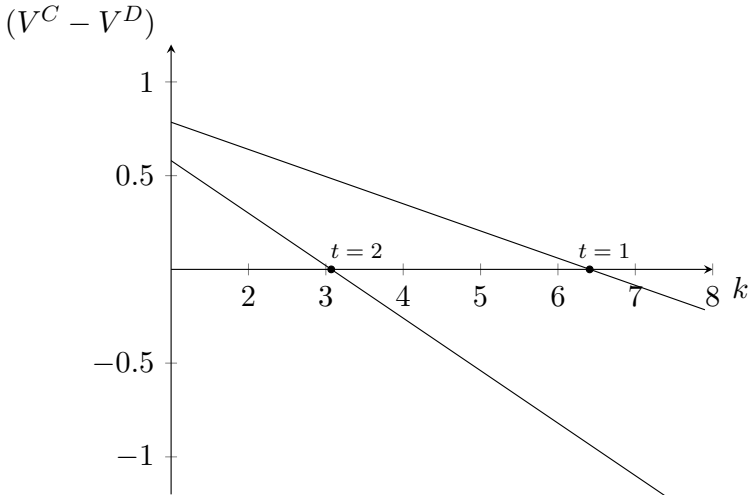


Figure 3: Effect of the efficiency multiplier k on the difference between chips' values $(V^C - V^D)$ varying heterogeneity parameters s^C, s^D, ρ^D

$t = 1$: $s^C = 0.9, s^D = 0.1, \rho^D = 0.05$;
$t = 2$: $s^C = 0.8, s^D = 0.2, \rho^D = 0.1$
$E^C = 1, \rho^C = 0.3$ for both $t = 1, 2$

As highlighted by the trilemma, the superiority of one chip over the other is based on three factors combined together. For example, it is not sufficient for a chip to exhibit the lowest energy consumption if the computing power and variety (heterogeneity) of calculations available are low. Moreover, even any pair–wise superiority can be outweighed by a deep enough inferiority with respect to the remaining third characteristic. In order to illustrate how the superiority of a chip is reached through the balancing of the three corner characteristic of the trilemma, we constructed a stylised example, visualised in Figure 3. In this example, in period $t = 1$ the

amount of tasks performed on a data–flow chip is only 10%, $s^D_{t=1} = 0.1$, and only 5% of tasks performed on a control–flow chip are interchangeably executable on the other, data–flow chip, $\rho^D_{t=1} = 0.05$. In period $t = 2$, both parameters are doubled, namely $s^D_{t=2} = 0.2$ and $\rho^D_{t=2} = 0.1$. Note that according to (3.3) if $s^D_{t=1} = 0.1$ hence $s^C_{t=1} = 0.9$, then if in the second period s^D increased to 0.2 hence $s^C_{t=2} = 0.8$. This doesn't necessarily mean that the absolute amount of tasks performed by the control–flow chip has shrunk; it might simply mean that the amount of tasks performed by the data–flow chip expanded without taking over tasks from the control–flow chip; this can be the case of ANNs. Thus, in period $t = 1$, given the setting, in order to have equal values, $(V^C - V^D) = 0$, for the data–flow chip it is required to be almost 6.5 times more efficient in terms of combined efficiency E (performance per Watt) than the control–flow chip, $k_{t=1} = 6.41$. In the second period $t = 2$, when the amount of software that can be run on the data–flow chip reaches 20%, $s^D = 0.2$, and interchangeability doubles from 5% to 10%, $\rho^D_{t=2} = 0.1$, in order to have equal values the data–flow chip has to be only 3 times more efficient, $k_{t=2} = 3.07$. This numerical example displays the mechanism at work, but might not represent an accurate calibration of the parameters. However, it illustrates the trade–offs and balance through which superiority can be achieved.

Finally, it is extremely important to note that, despite the fact that the numerical expressions of improvements of s^D and ρ^D are incremental from one time period to another, it can be rather technically hard to achieve such improvements which might also take a substantial amount of time; time periods used in the example are not specified but could be one year or a couple of years, resembling the timing of each next technological node, for example, of Moore's law. Furthermore, all relations in the model are linear, while in reality some nonlinearities can take place, however without drastically changing the main outcome of the outlined mechanism.

Supply Side. On the supply side we assume a duopoly with price competition. Thus, we are searching for a Nash–Bertrand equilibrium. By equalising utilities from equation (1) we obtain the indifferent consumer:

$$\hat{x} = \frac{V^C - V^D + p^D - p^C + \delta}{2\delta} \tag{7}$$

Each firm has a profit function:

$$\pi^C = p^C q^C = p^C \hat{x}; \quad \pi^D = p^D q^D = p^D (1 - \hat{x}) \tag{8}$$

Maximising profit with respect to price, we yield the response function:

$$\begin{aligned} &\max_{p^i} \pi^i \qquad \text{yielding} \\ R^C(p^D) = p^C &= \frac{V^C - V^D + p^D + \delta}{2} \\ R^D(p^C) = p^D &= \frac{V^D - V^C + p^C + \delta}{2} \end{aligned} \tag{9}$$

By replacing p^j in the response function $R^i(p^j)$ with $R^j(p^i)$ we derive the Nash–Bertrand pair:

$$p^C_{NB} = \frac{V^C - V^D + 3\delta}{3}; \quad p^D_{NB} = \frac{V^D - V^C + 3\delta}{3} \tag{10}$$

For equilibrium prices to be non–negative the condition $-3\delta \leq V^D - V^C \leq 3\delta$ has to be fulfilled.

Finally, plugging the results of equation (10) into (7) we obtain the final formula for the indifferent consumer:

$$\hat{x} = \frac{V^C - V^D + 3\delta}{6\delta} \tag{11}$$

The position of the indifferent consumer is defined by two factors. First, the difference in chips' values ($V^C - V^D$) whose analysis was shown in equations (4)–(6) can be employed here as well. Second, the disutility δ from the level of mismatch between computations used by a consumer and computations available on the purchased chip. Deepening further the interpretation, this means that the parameter δ reflects a degree of application specialisation or computational convergence. From a technological perspective, there are two extreme cases of relation between the terms *task* and *calculation type* and a whole continuum in between. The first corner case would be when every task is performed by a single type of calculation, which would establish an unequivocal correspondence between the terms task and calculation type; in that case δ is the highest. The second corner case would imply that every task consists of all possible types of calculations; in that case δ must be low. From an economic perspective, in the first case a user–firm runs only one type of calculation, hence it has a strong preference for a chip that runs this calculation better. More generally, if demand consists of consumers each employing highly homogeneous computation, δ for the chip–making industry would be very high. The realistic case is none of the extreme ones, with demand consisting of consumers each using its own mixture of types of calculations and only few consumers representing extreme cases using either purely homogeneous or purely heterogeneous calculations. That is why δ can be interpreted as a measure of mismatch between the heterogeneity supported on a chip and the heterogeneity of computations used by a consumer.

In sum, the price of a chip is higher, (i) the higher its overall value with respect to its competing alternative and (ii) the stronger the preference for a particular chip. With the simulated example illustrated in Figure 3 we have shown how the difference between chips' values varies when changing the chips' heterogeneity characteristics rooted in the software domain. The numerical results of this example with a fixed value of efficiency multiplier $k = 2$ and calculated equilibrium market shares, prices and profits is shown in Table 1. This simulation demonstrates the composite, "trade off" nature of chips' value as well as how the relative dominance of one chip over the other can be achieved; lack of heterogeneity supported by chip j can be compensated with higher combined efficiency E^j that comprises the other two vertices of the trilemma, computing speed and energy efficiency, and in doing so equalising the overall values of the chips.

The model doesn't contain a cost variable; however, implicitly a higher value is associated with higher costs to achieve it. For example, according to the financial statements of the ASML Holding (the leading company in the market of photolithography systems), the price of an average system sold in the first half of 2019 is in the range of 36–38 millions of euros (ASML Holding, 2019). Such equipment is highly standardised and it would only account for the initial part of the production process, not ensuring an increase in chip's value as the value of a chip is based on the trilemma's vertices. Leaving aside the formal mechanism of cost formation, in our model we deal with its final instantiation — the price. From an economic viewpoint, the

purpose of our model is to show how the shares of demand are driven by the characteristics of the trilemma and prices as a touch–point of supply and demand. Thus, costs are involved implicitly through the cost of production of a chip with a particular value and its improvement with respect to the characteristics of the trilemma. It is beyond our analysis to explain *how* a particular value of a chip is achieved, while extensive technological insights regarding chips' characteristics and directions of their improvement currently under exploration in the industry were provided in previous sections. Here we simply assume that every firm estimates its fixed costs to produce a chip and the quantity demanded in order to understand whether or not the production of a chip will be profitable, exploiting either a high price at low quantity or economies of scale at a low price. If a firm estimates that costs might overweight revenue, it doesn't enter the market.

Parallels in the Marketplace. Insofar as technological properties, being exogenous variables, are beyond formal modeling, they remain operational leverages on which a company can act in order to improve its product. At the end of Section 4.2 we described a decision–making process on the supply side according to which producers choose approach and degree of addressing the trilemma based on cost/benefit analysis, aspiring to create a product that appeals to a larger share of demand. This decision results in the next generation of products with different values offered to the consumers. In the previous sections we provided examples of innovations in architecture, elements, materials and techniques that target computing power, energy efficiency and heterogeneity. Hierarchical networks instead of bus interconnect, experimentation with wafer size, new materials and signal types, 3D instead of planar transistors, die stacking, in–memory computing, data flow and neuromorphic architectures among others, all illustrate producers' actions undertaken to address the trilemma from different sides and at different levels.

Provided that we model heterogeneity through the software domain, this implies that producers can allocate their effort to increase the heterogeneity supported by their products in two ways: (i) introducing changes in hardware to expand functionality and through that encompass more of the existing software from the competitor; in other words, increase ρ^i, (ii) investing in the expansion of the software set written specifically for a producer's own chip, which means increasing s^i. The first way, the introduction of hardware changes, is discussed at length in previous sections, therefore we now focus on the second way, software–related changes. As mentioned in Section 3.1 regarding GPGPU, Nvidia developed the CUDA framework to support its products; the consortium Khronos Group works in the same direction of heterogeneous computing with its OpenCL framework designed by Apple. Other open–source platforms like Google's TensorFlow and Microsoft's CNTK are aimed at the collaborative development of data–flow software solutions to run on chips that can support it such as TPU or CPU–GPU tandem. By adapting the existing software and writing programs that can effectively run on its product, a firm i increases the value of its chip i targeting precisely the s^i component. However, producers of the competing chip j can counteract by developing instruction set architecture (ISA) extensions.[26] Modifying ISA by including additional packages of new commands allows the competing chip j to encompass some functions performed on the chip i. In terms of our model, such effort

[26]ISA modifications in essence are on the borderline between hardware and software (programmatic) changes. Given ISA's undeniable programmatic element, here we employ example of ISA modifications that, similarly to the hardware changes, impact ρ^j.

affects ρ^j. As an example we can mention Advanced Vector Extensions (AVX) and its further extension Vector Neural Network Instructions (VNNI) from Intel for x86 ISA, Vector Multimedia Extension (VMX also known as AltiVec) by IBM for Power ISA and NEON technology from ARM Holdings for its eponymous ARM ISA.

In sum, our model reveals the mechanism driving demand distribution based on chips' technical characteristics, available software and how well overall a chip meets the computational needs of consumers. We want to stress that an increasing share of demand for alternative chips doesn't imply the extinction of the general–purpose one (the one representing the dominant design). It mainly highlights the idea that once the demand for alternative chips increases that sends a signal to producers that some function(s) in which alternative chips are specialised are growing in importance. Applied to the current situation in the semiconductor industry, the emergence of programs in the domain of AITs that successfully address new tasks and have a potential to proliferate further in both depth and scope increase the amount of tasks and importance of functions performed by data–flow chips. In terms of the model, this means that s^D grows while ρ^C remains low. The question is then whether or not these functions can be absorbed into an encompassing platform chip, and if yes how to achieve that through the series of steps we described when discussing the decision–making process related to chip development shaped by the trilemma. If that is not successful, the product loses the share of demand which uses this type of computation. For new alternative chips, success might be key to their very existence; for example, GPUs survived mainly due to the game industry, which was their persistent source of demand.

From our analysis it is becoming clear that the simple speedup race between competing types of chips is not the core issue for the future of the semiconductor industry; rather, the more profound issues of functions, computation types, and variety that a chip can support in order to appeal to a sufficient share of demand are key. This poses a serious challenge for the semiconductor industry that can be dealt with either trying to incorporate the required functions under a platform chip, or starting to produce a range of customised products. Both strategies are potential resolutions for the current crossroad situation at this point of the technological trajectory. The choice between the two strategies is guided by the trilemma from the technological side, with the set of directions under consideration in the industry we described in Section 3, and it is steered by the forces we highlighted in Section 4.2 from the economic side.

4.4 Scenarios and Discussion

From the discussion so far we set out a collection of mechanisms and forces shaping from the outside and within the evolution of the semiconductor industry. Exogenous challenges like the arrival of modern AITs test the robustness of the established technological trajectory. In fact, this might be the first time in which chips producers face a salient qualitative challenge instead of a quantitative one, that was solved in the past largely through miniaturisation of elements. The question now is how chips will evolve this time. Considering all the factors at play, we derived two scenarios on which the industry can converge.

Scenario I Under the relentless pressure of economic factors within the semiconductor in-

dustry and the continuous but siloed pull from the downstream markets for market–specific improvements, producers might decide to pursue trajectories tailor–made to subsets of downstream markets, grouped around specialised chips that accurately address needs within given submarkets. A *customisation strategy and hence the fragmentation of the semiconductor industry* might occur.

Scenario II Aspiring to address larger shares of demand associated with greater but probably delayed payoffs, chip producers can make long–term investments at the system level, aimed at the creation of a *platform chip comprising heterogeneous cores, resulting in a new dominant design*. To achieve that, the overarching architecture must reproduce a composition of components on a chip that ensures scalable, heterogeneous and energy–proportional computing. The inherent modularity of such architecture would allow the product to adapt to any potential combination of consumers' needs and business models. Developed in response to the call of one industry, the platform chip can diffuse over time among other downstream markets with decreasing cost of production and, hence, price.

In previous sections we discussed what can make the semiconductor industry to remain on a persistent track or to derail from it. The industry–wide exploration provoked by modern AITs results in novel products that obtain positive feedback from demand, in a manner captured by the model in Section 4.3. Eventually, this renewed turbulence in the industry will end up in one of the two scenarios outlined above. Arguments "pro" and "against" exists for both of them. For example, Thompson & Spanuth (2018) advocate for the first scenario by linking the future dynamics of chips production to the dual–inducement mechanism typical of General Purpose Technologies (GPTs) (Bresnahan & Trajtenberg, 1995). They develop a model of choice between universal and specialised processors based on relative speed up factor and identify a cut–off point from which the specialised processors become more appealing than the universal ones. As more and more downstream markets switch to specialised processors, this leads to the halt of the dual–inducement mechanism for universal processors. Thus, they expect the end of the GPT paradigm of universal processors and envisage a situation of application–based market fragmentation with specialised computing evolving in more compartmentalised domains. This prediction rests on a view of the processor as the singleton GPT technology; processors can, from this perspective, be pure competing alternatives. However, we also need to consider the possibility that it is the SoC the candidate for the role of GPT — as it is the whole system, rather than the sole processor, to be used for downstream applications and innovation. As Borkar & Chien (2011, p.75) write: "[t]he role of microprocessor architect must expand beyond the processor core, into the whole platform on a chip, optimising the cores as well as the network and other subsystems".

Undoubtedly, modern AITs are changing the balance between the amount of tasks performed on CPU and what used to be, for example, graphics accelerator. Some functions used by modern AITs cannot be performed (effectively or at all) on a chip based purely on the von Neumann architecture. The more AI penetrates businesses and everyday life of individual consumers, essentially just like computers or the Internet did, the more common and ubiquitous the hardware that supports it might become. Thus, what was outsourced to an accelerator might become part

of a regularly required functionality, making the accelerator a compulsory logical block of the chip. In this sense, rather than its substitution, we witness the evolution of a GPT; the GPT is the SoC, however internally evolving in architecture and design to remain a universal computing device applicable to the widest range of tasks.

Besides speedup, the other important factor to take into account is the heterogeneity of computation available on the chip. Given that heterogeneity can be approximated with the amount of software that is effectively run on the chip, the presence of indirect network externalities does have significant implications for the semiconductor industry. In favor of the second scenario, this approach suggests that consumers' decision upon which hardware system to buy is affected by complementary products or supporting services available for each system. In particular, Church & Gandal (1992) model the effect of the decision of software firms upon software provision on the market share and the number of hardware systems that will exists in equilibrium. Their analysis shows that when consumers' preferences on *software variety* are relatively high[27], this leads to the exclusive adoption of one of the hardware systems if a critical minimum amount of software is provided. Furthermore, in the case in which two hardware systems exist, total surplus would be higher under many parameters' values if a standard (a single hardware system) was mandated. Thus, strong preference for software variety is associated with the choice of one hardware system. By translating software variety into hardware's supported heterogeneity, in our model heterogeneity acts as a variable that characterises the chip; a producer can increase the value of its chip either equalising it with the competitor's one or surpassing it as shown in the example represented in Figure 3. Lastly, preference for software variety and hence hardware heterogeneity is consistent with the progress in the field of AI and its effort to emulate human intelligence in virtual or physical machines (recall the quote of Minsky about diversity as a root of intelligence in the end of Section 2.2).

The deployment of modern AI systems involves an massive amount of computation, and as shown in the Open AI report this also doubles every 3.4 months since 2012 (Amodei et al., 2019). However, the report specifically stipulates that "[m]assive compute is certainly not a requirement to produce important results" and provides a set of examples supporting this statement. An exploration into this direction is conducted by an entire thread of research dedicated to the development of techniques of model compression for modern AITs (Cheng et al., 2017; Lin et al., 2019). First, model compression could make more feasible to deploy AI systems on devices with limited resources like energy, memory, CPU, bandwidth, etc. A transfer of processing away from the servers of tech giants and towards the edge may significantly contribute to the discussion on privacy and data ownership (see, for example, Chiou & Tucker (2017), Acquisti et al. (2016)). Second, but more important, the development and training of modern AITs that demonstrate record–breaking magnitudes of compute has an enormous environmental toll. Training and further fine–tuning of network connections (neural architecture search) are the most expensive in terms of computing and hence energy consumption, resulting in substantial CO_2 emissions. In general, state–of–the–art models do not remain uncontested for a long time and evolve into updated computationally more hungry versions, however delivering sometimes marginal improvement at inadequate costs. Combining the Open AI report on amount of com-

[27] The benefit from high software variety available on a chosen hardware system has to overweight the disutility from spending on the purchase of various software, the price of hardware and the degree of its differentiation.

pute and its unprecedented 300000–fold increase since 2012 with estimations done in Strubell et al. (2019), the looming perspective is alarming. This strongly calls for energy efficient software and hardware solutions.

The growing amount of compute creates the misleading impression that faster hardware is one–size–fits–all solution. Instead, already 60 years ago, at the dawn of AI, researchers concluded that "the speeds and memory capacities of present computers may be insufficient to simulate many of the higher functions of the human brain, but the major obstacle is not lack of machine capacity but our inability to write programs taking full advantage of what we have." (McCarthy et al., 1955). Now, in 2019, this fundamental conclusion remains unchanged: "[f]ocusing on raw computing power misses the point entirely. Speed alone won't give us AI. Running a poorly designed algorithm on a faster computer doesn't make the algorithm better; it just means you get the wrong answer more quickly. (And with more data there are more opportunities for wrong answers!) The principal effect of faster machines has been to make the time for experimentation shorter, so that research can progress more quickly. It's not hardware that is holding AI back; it's software." (Russell, 2019). Thus, despite the fact that currently leading AI techniques exhibit high magnitudes of compute and hence require faster hardware, it seems a short–term–looking strategy for the semiconductor industry to abandon development of a general purpose chip as the basis for the technological trajectory and switch completely to the customised, extremely fast but homogeneous hardware, especially because multiple predictions for the field of AI itself anticipate a variety of approaches.

Finally, we would like to stress that our analysis of the AI–triggered transformation of the semiconductor industry was largely focused on the demand side which is driven by the technological characteristics of chips. However, it is important to highlight an additional level of complexity related to the oligopolistic nature of the semiconductor industry. The pull towards one or the other scenario is exerted not only by the demand side (downstream markets) but, given the high concentration of market power among chip producers, also by the supply side. The dominant digital giants (platforms) have an incentive to centralise computation around themselves, pursuing the goal of exploiting dynamic returns from data accumulation and data feedback loops. The centralisation of computation is enabled by facilitating and advertising cloud–based services and pushing for "thin client" business model with consumers' hardware being merely an access device to massive computing facilities in the cloud. This case bears a remarkable resemblance with the mainframe–terminal computing paradigm preceding the advent of personal computers (PC). If this nudging takes place and appears successful in the future, the semiconductor industry indeed might be transformed into a "fast line" and a "slow line" of production, where the first one provides expensive chips for cloud computing facilities of dominant platforms and the second one manufactures the terminal "thin client" chips. The estimation of welfare distribution under such conditions is yet to come.

Concerning the implications for the supply side, if the "thin client" strategy works out, among the semiconductor industry players there will be losers and winners too. Back in the past, the partnership between Microsoft and Intel, dubbed Wintel, created a strong lock–on situation through the production of complementary products, Microsoft's Windows operating system (OS) and Intel's chips (Casadesus-Masanell & Yoffie, 2007). The mutual reinforcement

cycle between Microsoft's OS improvements (software) optimised for Intel's chips (hardware) and the subsequent Intel's hardware improvements fruitfully exploited by Microsoft's OS maintained the dominance of Wintel's joint product over decades. Nowadays, a new lock–on situation might arise from current dominant digital platforms whose evident effort to create in–house chips to support proprietary cloud services (for examples see Section 3.1) points at a conclusion that a new Wintel might be on its way. There is an opportunity though for the semiconductor industry and downstream markets to derail from the new Wintel track. For the demand side, there are alternative options such as federated computing and edge computing which by nature advocate for a "thick client" business model. With the decreasing cost of advanced digital technologies over time, the realisation of the "thick client" model is getting more affordable for the demand while also allowing to retain data ownership, control over processes and supporting systems. Therefore, the high–end chips, from being an expensive privilege of the dominant digital platforms, can turn over time into products available for a wide range of downstream markets. If that happens, it would complete a new swing in a longer technological cycle similar to the transition from the mainframe to the PC while this time from cloud to edge computing. For the supply side, furthering computation at the edge can be also a strategy to pursue for companies trying to avoid harsh competition with dominant digital platforms. In sum, the demand–pull for thicker clients and devices capable of computing at the edge will stress other dimensions on which to focus attention and allocate efforts and resources. The outcome of this competition between higher level technological systems and business models is yet to be seen, but will reverberate and layer–up complexity to the dynamics of the semiconductor industry we unpacked in our analysis.

5 Conclusion

In this paper we conducted a guided–tour through the techno-economic dimensions that affect the architectural and design choices for integrated circuits. The analysis uncovers crucial differences of feasibility and efficiency of tasks' solution among differently organised computing devices. The primary appearance of the von Neumann architecture imposed the manner of formulation and hence solution of many tasks; indeed, the formulation of many tasks in terms of control flow logic could have been a choice forced by the ubiquity of the von Neumann architecture, rather than driven by the efficiency of an algorithm written in that manner. However, new approaches in computation are gaining momentum, inducing a new wave in the life cycle of the semiconductor industry. While analysing that, this paper turned out to be a study of a technological system consisting of two big components: hardware and software. This technological system exhibits circular connections between its components; a shock in the software component with the success of modern AITs, and ANNs in particular, reverberates to the hardware component in form of a challenge to incorporate into devices the needs of the novel approach, whose results, in turn, will feedback on software and so on. In the midst of this cumulative causation, the established technological trajectory in the semiconductor industry is shaken and contested, and new turbulence is injected as alternative paths to follow become viable and different techno–economic constraints to production become binding.

Among the identified characteristics of a chip, which are of interest for the demand and

relevant to product development for the supply, heterogeneity (variety of software supported on a chip) plays a salient role but technically it is getting harder to achieve given the expansion of algorithms' variety. The core issue at stake now is the fate of the current technological trajectory, which will either fork into multiple architectures each serving a smaller set of downstream markets or will be rejuvenated by the arrival of a new dominant design in form of platform (general purpose) chip efficiently supporting heterogeneous computing. One cannot rule out that during the exploration of new architectures and designs, the industry can exhibit multiple product versions; the distinction between this situation and true fragmentation is the persistence of such state over time. The current explosion of novel chips reminds an exploration phase rather than a persistent fragmentation and the end of the general purpose chip era.

Eventually, the heterogeneity supported by a chip maps onto applications and economic activities built around them. A growing amount of businesses is increasingly using modern AITs, especially dominant digital platforms, given their interest in predictive tasks which are by nature probabilistic and require data; this interest is not a coincidence as only some share of tasks can enjoy a deterministic, full–certainty property. More complex and less structured tasks can be a better fit for the application of probabilistic estimations and statistical methods which made viable by data abundance. Modern AITs based on such methods expanded the set of applications of computations as a problem–solving tool. Currently, the semiconductor industry is in search of an adequate response to this expansion. The broader the scope of adoption of modern AITs among downstream markets, the more stronger the need for this response.

Finally, a prospective direction for further research could be to investigate the composition of demand preferences over the trilemma's characteristics. An estimation of the characteristics' relative importance could provide a hint into which direction the technological trajectory is more likely to be pulled by the demand side. At the same time, the supply side might have a different vision. This crossroad in the semiconductor industry, occasionally or not, coincided with the trend of cloud computing and the aspiration of digital platforms to set users on a "thin client" model. To know whether the preferences of demand and supply converge or diverge would greatly help the estimation of the likelihood of our scenarios.

References

Acquisti, A., Taylor, C., & Wagman, L. (2016). The economics of privacy. *Journal of Economic Literature*, 54(2), 442–92.

Agrawal, A., Gans, J. S., & Goldfarb, A. (2019). *Artificial Intelligence: The Ambiguous Labor Market Impact of Automating Prediction*. Technical report, National Bureau of Economic Research.

Amdahl, G. M. (1967). Validity of the single processor approach to achieving large scale computing capabilities. In *Proceedings of the April 18-20, 1967, spring joint computer conference* (pp. 483–485).: ACM.

Amodei, D., Hernandez, D., Sastry, G., Clark, J., Brockman, G., & Sutskever, I. (2019). Ai and compute.

Ardila, D., Kiraly, A. P., Bharadwaj, S., Choi, B., Reicher, J. J., Peng, L., Tse, D., Etemadi, M., Ye, W., Corrado, G., et al. (2019). End-to-end lung cancer screening with three-dimensional deep learning on low-dose chest computed tomography. *Nature Medicine*.

Arthur, W. B. (1989). Competing technologies, increasing returns, and lock-in by historical events. *The economic journal*, 99(394), 116–131.

ASML Holding, N. (2019). Financial statements us gaap q2 2019.

Auth, C., Allen, C., Blattner, A., Bergstrom, D., Brazier, M., Bost, M., Buehler, M., Chikarmane, V., Ghani, T., Glassman, T., et al. (2012). A 22nm high performance and low-power cmos technology featuring fully-depleted tri-gate transistors, self-aligned contacts and high density mim capacitors. In *2012 Symposium on VLSI Technology (VLSIT)* (pp. 131–132).: IEEE.

Belleflamme, P. & Peitz, M. (2018). 11. platforms and network effects. *Handbook of Game Theory and Industrial Organization, Volume II: Applications*, 2, 286.

Boden, M. A. (2016). *AI: Its nature and future*. Oxford University Press.

Bohr, M. T., Chau, R. S., Ghani, T., & Mistry, K. (2007). The high-k solution. *IEEE spectrum*, 44(10), 29–35.

Borkar, S. & Chien, A. A. (2011). The future of microprocessors. *Communications of the ACM*, 54(5), 67–77.

Bresnahan, T. F. & Trajtenberg, M. (1995). General purpose technologies 'engines of growth'? *Journal of econometrics*, 65(1), 83–108.

Cabral, L. (2018). *Standing on the Shoulders of Dwarfs: Dominant Firms and Innovation Incentives*. Technical report, CEPR Discussion Papers.

Cantner, U. & Vannuccini, S. (2017). 11. innovation and lock-in. *The Elgar Companion to Innovation and Knowledge Creation*, (pp. 165).

Casadesus-Masanell, R. & Yoffie, D. B. (2007). Wintel: Cooperation and conflict. *Management Science*, 53(4), 584–598.

Chen, Y.-H., Yang, T.-J., Emer, J., & Sze, V. (2019). Eyeriss v2: A flexible accelerator for emerging deep neural networks on mobile devices. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*.

Cheng, Y., Wang, D., Zhou, P., & Zhang, T. (2017). A survey of model compression and acceleration for deep neural networks. *arXiv:1710.09282*.

Chiou, L. & Tucker, C. E. (2017). Search engines and data retention: Implications for privacy and antitrust. *NBER Working Paper*.

Chou, C.-f. & Shy, O. (1993). Partial compatibility and supporting services. *Economics letters*, 41(2), 193–197.

Church, J. & Gandal, N. (1992). Network effects, software provision, and standardization. *The journal of industrial economics*, (pp. 85–103).

Cuesta, F. A., Sequeiros, P. G., & Rojo, Á. L. (2017). A method for validating rent's rule for technological and biological networks. *Scientific reports*, 7(1), 5378.

David, P. A. (2007). Path dependence: a foundational concept for historical social science. *Cliometrica*, 1(2), 91–114.

Davies, M., Srinivasa, N., Lin, T.-H., Chinya, G., Cao, Y., Choday, S. H., Dimou, G., Joshi, P., Imam, N., Jain, S., et al. (2018). Loihi: A neuromorphic manycore processor with on-chip learning. *IEEE Micro*, 38(1), 82–99.

Dennard, R. H., Gaensslen, F. H., Rideout, V. L., Bassous, E., & LeBlanc, A. R. (1974). Design of ion-implanted mosfet's with very small physical dimensions. *IEEE Journal of Solid-State Circuits*, 9(5), 256–268.

Dosi, G. (1982). Technological paradigms and technological trajectories: a suggested interpretation of the determinants and directions of technical change. *Research policy*, 11(3), 147–162.

Eckersley, P., Nasser, Y., et al. (2017). Eff ai progress measurement project.

Flamm, K. (2018). *Measuring moore's law: Evidence from price, cost, and quality indexes*. Technical report, National Bureau of Economic Research.

Goldfarb, A., Gans, J., & Agrawal, A. (2019). *The Economics of Artificial Intelligence: An Agenda*. University of Chicago Press.

Greenstein, S. (2019). Digital infrastructure. In *Economics of Infrastructure Investment*. University of Chicago Press.

Hebb, D. O. (2005). *The organization of behavior: A neuropsychological theory*. Psychology Press.

Henderson, R. M. & Clark, K. B. (1990). Architectural innovation: The reconfiguration of existing. *Administrative science quarterly*, 35(1), 9–30.

Hennessy, J. L. & Patterson, D. A. (2011). *Computer architecture: a quantitative approach*. Elsevier.

Jouppi, N. P., Young, C., Patil, N., Patterson, D., Agrawal, G., Bajwa, R., Bates, S., Bhatia, S., Boden, N., Borchers, A., et al. (2017). In-datacenter performance analysis of a tensor processing unit. In *2017 ACM/IEEE 44th Annual International Symposium on Computer Architecture (ISCA)* (pp. 1–12).: IEEE.

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems* (pp. 1097–1105).

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *nature*, 521(7553), 436–444.

Lee, E. A. (2002). Embedded software. In *Advances in computers*, volume 56 (pp. 55–95). Elsevier.

Lin, J., Gan, C., & Han, S. (2019). Tsm: Temporal shift module for efficient video understanding. In *Proceedings of the IEEE International Conference on Computer Vision* (pp. 7083–7093).

Liu, G., Ahsan, S., Khitun, A. G., Lake, R. K., & Balandin, A. A. (2013). Graphene-based non-boolean logic circuits. *Journal of Applied Physics*, 114(15), 154310.

McCarthy, J., Minsky, M., Rochester, N., & Shannon, C. (1955). Proposal for the 1956 dartmouth summer research project on artificial intelligence, dartmouth college.

McCulloch, W. & Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, 5, 115–133.

Merolla, P. A., Arthur, J. V., Alvarez-Icaza, R., Cassidy, A. S., Sawada, J., Akopyan, F., Jackson, B. L., Imam, N., Guo, C., Nakamura, Y., et al. (2014). A million spiking-neuron integrated circuit with a scalable communication network and interface. *Science*, 345(6197), 668–673.

Moore, G. E. (1965). Cramming more components onto integrated circuits.

Newell, A., Shaw, J. C., & Simon, H. A. (1959). Report on a general problem solving program. In *IFIP congress*, volume 256: Pittsburgh, PA.

Newell, A. & Simon, H. (1956). The logic theory machine–a complex information processing system. *IRE Transactions on information theory*, 2(3), 61–79.

Page, S. E. (2006). Path dependence. *Quarterly Journal of Political Science*, 1(1), 87–115.

Perrault, R., Shoham, Y., Brynjolfsson, E., Clark, J., Etchemendy, J., Grosz, B., Lyons, T., Manyika, J., Mishra, S., & Niebles, J. C. (2019). *The AI Index 2019 Annual Report*. Technical report, AI Index Steering Committee, Human-Centered AI Institute, Stanford University.

Raina, R., Madhavan, A., & Ng, A. Y. (2009). Large-scale deep unsupervised learning using graphics processors. In *Proceedings of the 26th annual international conference on machine learning* (pp. 873–880).: ACM.

Russell, S. (2019). *Human Compatible: Artificial Intelligence and the Problem of Control*. VIKING.

Schwierz, F. (2010). Graphene transistors. *Nature nanotechnology*, 5(7), 487.

Selfridge, O. G. (1958). Pandemonium: A paradigm for learning. *the Mechanisation of Thought Processes*.

Shannon, C. E. (1948). A mathematical theory of communication. *Bell system technical journal*, 27(3), 379–423.

Shen, Y., Harris, N. C., Skirlo, S., Prabhu, M., Baehr-Jones, T., Hochberg, M., Sun, X., Zhao, S., Larochelle, H., Englund, D., et al. (2017). Deep learning with coherent nanophotonic circuits. *Nature Photonics*, 11(7), 441.

Shy, O. (2011). A short survey of network economics. *Review of Industrial Organization*, 38(2), 119–149.

Simon, H. A. (2002). Near decomposability and the speed of evolution. *Industrial and corporate change*, 11(3), 587–599.

Steinmueller, W. E. (1992). The economics of flexible integrated circuit manufacturing technology. *Review of Industrial Organization*, 7(3-4), 327–349.

Strubell, E., Ganesh, A., & McCallum, A. (2019). Energy and policy considerations for deep learning in nlp. *arXiv:1906.02243*.

Su, L. (2019). Delivering the future of high-performance computing with system, software and silicon co-optimization, keynote address at hot chips: A symposium on high performance chips, edition 31.

Suárez, F. F. & Utterback, J. M. (1995). Dominant designs and the survival of firms. *Strategic management journal*, 16(6), 415–430.

Thompson, N. & Spanuth, S. (2018). The decline of computers as a general purpose technology: Why deep learning and the end of moore's law are fragmenting computing. *SSRN 3287769*.

Turing, A. M. (1937). On computable numbers, with an application to the entscheidungsproblem. *Proceedings of the London mathematical society*, 2(1), 230–265.

Veen, A. H. (1986). Dataflow machine architecture. *ACM Computing Surveys (CSUR)*, 18(4), 365–396.

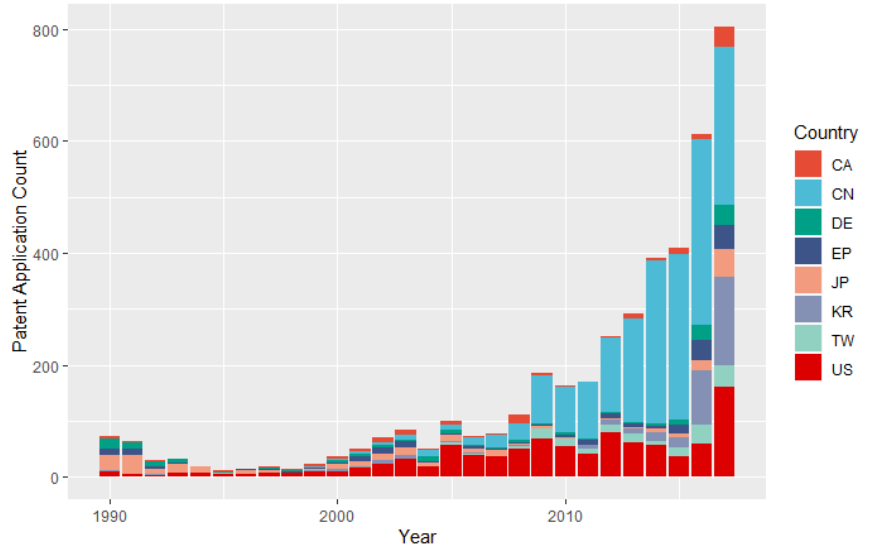
Weiser, M. (1999). The computer for the 21st century. *ACM SIGMOBILE mobile computing and communications review*, 3(3), 3–11.

Winter, M., Prusseit, S., & Gerhard, P. F. (2010). Hierarchical routing architectures in clustered 2d-mesh networks-on-chip. In *2010 International SoC Design Conference* (pp. 388–391).: IEEE.

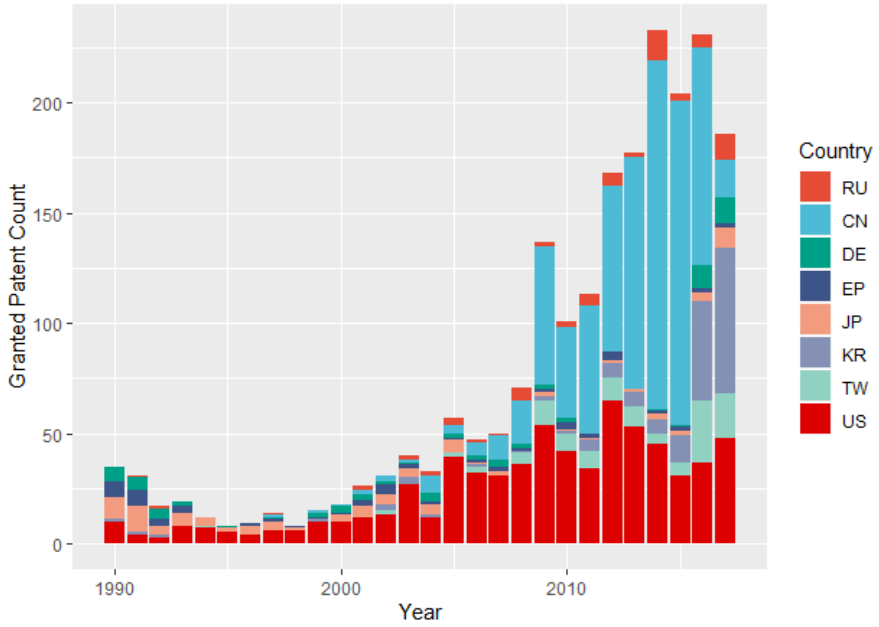
WIPO (2019). Wipo technology trends 2019 artificial intelligence.

Appendices

Appendix A Distribution of patent applications across patent authorities



(a) Filed patent applications across patent authorities



(b) Granted patents across patent authorities

Figure 4: Breakdown of the aggregate application dynamics by patent authorities with the highest total number of (a) filed applications and (b) granted patents over 1990–2017 period
Data: PATSTAT Global 2019 Autumn

Appendix B Continuation of the numerical example with two time periods and fixed $k = 2$

$$t = 1\text{: } s^C = 0.9, s^D = 0.1, \rho^D = 0.05;$$
$$t = 2\text{: } s^C = 0.8, s^D = 0.2, \rho^D = 0.1$$
$$E^C = 1, k = 2, \rho^C = 0.3, \delta = 0.3 \text{ for both } t = 1, 2$$

$t = 1$	$t = 2$
$V^C = 0.93, V^D = 0.29$	$V^C = 0.86, V^D = 0.56$
$p^C = 0.513, p^D = 0.086$	$p^C = 0.4, p^D = 0.2$
$\hat{x} = 0.85$	$\hat{x} = 0.66$
$\pi^C = 0.436, \pi^D = 0.013$	$\pi^C = 0.264, \pi^D = 0.068$

Table 1: Prices, profits and demand shares calculations

Recent papers in the SPRUWorking Paper Series:

December

2019-25. Mobilizing the Transformative Power of the Research System for Achieving the Sustainable Development Goals. *Matias Ramirez, Oscar Romero, Johan Schot and Felber Arroyave*

November

2019-24. Job Composition and Its Effect on UK Firms in the Digital Era. *Mabel Sánchez Barrioluengo*

2019-23. Start-up Subsidies: Does the Policy Instrument Matter? *Hanna Hottenrott and Robert Richstein*

2019-22. Organised Crime and Technology. *Mustafa Caglayan, Alessandro Flamini and Babak Jahanshahi*

October

2019-21. The Value of Data: Towards a Framework to Redistribute It. *Maria Savona*

September

2019-20. Teaming up with Large R&D Investors: Good or Bad for Knowledge Production and Diffusion? *Sara Amoroso and Simone Vannuccini*

2019-19. Experimental Innovation Policy. *Albert Bravo-Biosca*

August

2019-18. Relating Financial Systems to Sustainability Transitions: Challenges, Demands and Dimensions. *Chantal P. Naidoo*

Suggested citation:
Ekaterina Prytkova and Simone Vannuccini (2020). On the Basis of Brain: Neural-Network-Inspired Change in General Purpose Chips. SPRU Working Paper Series (SWPS), 2020-01: 1-47. ISSN 2057-6668. Available at: www.sussex.ac.uk/spru/swps2020-01

Science Policy Research Unit
University of Sussex, Falmer
Brighton BN1 9SL
United Kingdom

SPRU website: **www.sussex.ac.uk/business-school/spru**
SWPS website: **www.sussex.ac.uk/business-school/spru/research/swps**
Twitter: **@spru**

UNIVERSITY OF SUSSEX | BUSINESS SCHOOL