Homework [Extra Credit] Assignment 8

1. Please read the ISSCC 2014 Keynote Publication by Professor Mark Horowitz "Computing's Energy Problem (and what we can do about it)"

1.1 How does Technology Scaling decrease the cost of Computing? How do reductions in the cost of manufacturing a transistor enable widespread use of computing devices?

1.1)

Moore's Law's embodiment in dimensional scaling of Integrated Circuits has dominated the semiconductor industry for the past 50 years (ICs). Moore's law states that every two years, the number of transistors in a dense integrated circuit (IC) doubles. Moore's law is a historical tendency that has been observed and projected. It is an empirical relationship tied to benefits from production experience, rather than physical law. As the cost of computing power declines for consumers, the cost for manufacturers to meet Moore's law rises: R&D, manufacturing, and testing costs have gradually climbed with each new generation of chips. Rising manufacturing costs are a major factor in Moore's law's continuation.

The economic counterpart to Moore's (first) rule, which states that the number of transistors in a dense integrated circuit doubles every two years, is Rock's law. The latter is a direct result of the capital-intensive semiconductor industry's continued expansion: more profitable and popular goods mean more capital available to spend in ever higher degrees of large-scale integration, which leads to the creation of even more revolutionary products.
Another design abstraction is that of a computer. It illustrates a transistor circuit whose operation is dictated by instructions read from an associated memory. Once complicated integrated circuits were available, it became possible to incorporate a small computer onto a single IC, resulting in a "micro-computer" or "microprocessor." This architecture made it much easier to make use of low-cost transistors; new applications could be created by simply modifying the instructions sent to an existing microprocessor to get the desired result. The simplicity of building and delivering computer-based solutions, together with the falling cost of computation, boosted demand for this device significantly.

New computing techniques, such as quantum computing, that alter the fundamental building elements of a computer will necessitate the development of new abstraction layers, software, and design frameworks to allow designers to construct and utilize these systems as their complexity grows. For new technologies, the expenses of developing these new hardware and software tools are critical, because the price of early machines will need to be high enough to recoup part of the costs. When competing against a seasoned player, this premium constantly punishes fresh methods.

1.2 Why did scaling processor clock frequency become more difficult in the last 15 years? How did Power dissipation become the primary constraint on server CPU performance?

1.2)

The transistors, which operate as switches to build logical gates, are the most crucial components of a processor. Our CPUs' hardest-working components are logical gates. They constitute units capable of arithmetic and complicated logical operations when put together in various configurations. In layman's words, the frequency at which a transistor can transition from on to function without failure is restricted. Because transistors are the building blocks of logical gates, our processor's working speed is similarly constrained by the switching frequency.

The processor's clock speed (also known as the clock frequency or clock rate) is a form of certification that tells us how often we can feed instructions while still getting reliable results. In the opposite direction, a processor with a 3 GHz clock speed can do 3 billion operations per second and still work as expected. We can accomplish more work per unit time if we have more operations per second. This implies that the user's computer programs will operate quicker – without requiring any code changes.

A processor's transistor count is the number of transistors it has. Because CPUs are generally the same size, the transistor count is proportional to their size. We can observe that transistor sizes have reduced dramatically from 170 million transistors in an Intel® Pentium® 4 CPU in 2004 to 4.3 billion transistors in a 15-core Intel Xeon Ivy Bridge processor in 2013. Moore noted this trend and detailed it in his law, which claims that transistor integration density doubles every 18 to 24 months ("The Next Generation of Moore's Law").

Another key finding is Dennard scaling, which states that the amount of power required to drive the transistors in a given unit volume remains constant as the number of transistors increases, resulting in voltage and current scaling with length. However, as transistors become smaller, this observation is becoming obsolete. The voltage and current scaling with length is nearing its limitations, as transistor gates have gotten too thin, compromising their structural integrity, and currents have begun to leak.
Furthermore, thermal losses occur when you are putting several billions of transistors together on a small area and switching them on and off again several billion times per second. The faster we switch the transistors on and off, the more heat will be generated. Without proper cooling, they might fail and be destroyed. One implication of this is that a lower operating clock speed will generate less heat and ensure the longevity of the processor. Another severe drawback is that an increase in clock speed implies a voltage increase and there is a cubic dependency between this and the power consumption. Power costs are an important factor to consider

when operating computing centers.
A designer must evaluate the cost-benefit trade-offs of all design options before selecting the features and parameter settings that give the highest performance per unit of energy. The huge number of processor design alternatives and the necessity to balance trade-offs in architecture, circuit design, and perhaps technology has made this technique challenging to automate in practice.

1.3 Why is Moore's Law slowing down? Why did Dennard Scaling end?

1.3)

Moore's Law has been kept alive by the chip industry, with companies like Intel, AMD, and IBM, to mention a few, developing a variety of microprocessor optimizations to translate the continual supply of new transistors into increased performance. This has had a major influence on the computer industry as well as society at large. As transistors have decreased, computers have gotten more powerful, culminating in a surge of advancements that have altered our daily lives. Take, for example, cellphones, high-speed Internet, and scientific advancements in sectors like weather and climate prediction, AI, genetics, and medical research.

The complexity of semiconductor process technologies has always grown. Moore's law has been fueled by this "innovation engine." The rise in complexity has been rising in recent years. Transistors are now three-dimensional electronics with odd behaviour. To precisely recreate these characteristics on a silicon wafer, sophisticated process technologies require numerous exposures (multi-patterning). The design process has become significantly more complicated as a result.

Moore's law has been hampered by all of this intricacy. Moving to a new process node is still a possibility, but the high cost and difficulty of doing so have delayed migration. Furthermore, each new manufacturing node currently yields less significant density, performance, and power reduction outcomes.

Moore's law and Dennard scaling were synonymous for many years. Supply voltage and current should be proportional to the linear dimensions of a transistor, according to Robert Dennard 1974. As a result, power became proportional to the transistor area as transistors shrunk. To put it another way, the power density did not change. We could, on the other hand, run the transistors at greater frequencies while maintaining the same power. Unfortunately, we can no longer grow supply voltage without raising leakage current exponentially in recent semiconductor technologies.

1.4 Why is the energy consumption by Memory substantial?

1.4)

The cache memory is a small, high-speed memory, which is designed for Static RAM (SRAM) and consists of the most recently accessed data of the main memory. Due to their size, caches cannot store all the code and data of an executing program. The cache memory is situated between the processor and the main memory Dynamic RAM (DRAM). Caches are known to perform 75% faster than their DRAM counterparts [10]. This is because it takes a shorter amount of time (15 ns) to retrieve information stored in the cache memory than the DRAM (60 ns). Moreover, the process of fetching an instruction from storage consumes time and power; therefore, to avoid the performance bottleneck at the input, the cache needs to be fast.

The memory design strictly centers on the principle of locality reference, meaning that at any given time, the processor accesses a small or localized region of memory. The cache loads this localized region. The internal 16 K byte cache of a Pentium processor contains over 90% of the addresses requested by the processor, making a hit rate of 90%. It is not feasible to replace the main memory with SRAM to upgrade the performance because it is very expensive, less dense, and consumes more power than DRAM. Therefore, increasing the amount of SRAM will hurt the performance since the processor will have more area to search, thus resulting in more time and dynamic power being spent on fetching. In addition, the cache needs to be of a size that the processor can quickly determine a hit or a miss to avoid performance degradation.

The power consumption of the Complementary Metal Oxide Semiconductor (CMOS) in CMPs is either dynamic power or leakage power. Dynamic power dissipation occurs in CMOS whenever transistors switch to change in a particular mode. This happens when the processor is either fetching for instruction, reading, or writing. Leakage power on the other hand occurs because of device inactivity or standby mode.

The techniques to save dynamic power do so by adjusting the voltage and frequency operation or by reducing the activity factor; whereas the techniques aiming to reduce the leakage power consumption seek to re-design the circuit to use low power cells and also reduce the total number of transistors or by putting the unused parts of the cache into low or sleep leakage mode.

1.5 What solutions to Computing’s Energy Problem does Professor Mark Horowitz’s envision?

1.5)

Professor Mark Horowitz’s envisions:

If technology is scaling more slowly

- We can incorporate current design knowledge into tools
- To create an extensible system constructor

If killer products are going to be application-driven

- Application experts need to design them

Leverage the first bullet point to enable the second

Over the last decades, the general-purpose computing stack and its abstractions have provided both performance and productivity, which have been the main drivers for the revolutionary advances in the IT industry. However, the computational demand for emerging applications grows rapidly, and
the rate of data generation exceeds the level where the capabilities of current computing systems can match. The challenges have coincided with the Dark Silicon era in which conventional technologies offer insufficient performance and energy efficiency. Thus, it is time to move beyond conventional techniques and explore radical approaches that can overcome the limitations of general-purpose systems and deliver large gains in performance and efficiency. One such approach is specialization, where the hardware and systems are developed for a domain of applications.

However, the specialization creates a tension between performance and productivity since programmers need to delve into the details of specialized hardware and perform low-level programming. Hence, the objectives are to deliver large gains in performance and efficiency while retaining automation and productivity through high-level abstractions. Achieving both conflicting objectives is a crucial challenge to place the specialization techniques in a position of practical utility.

A growing number of commercial and enterprise systems increasingly rely on compute-intensive Machine Learning (ML) algorithms. Hardware accelerators offer several orders of magnitude higher performance than general-purpose processors and provide a promising path forward to accommodate the needs of ML algorithms. Even software companies have begun to incorporate various forms of accelerators in their data centers. Microsoft's Project Brainwave integrated FPGAs on a data center scale for real-time AI calculations and Google developed the TPU as a specialized matrix multiplication engine for machine learning. However,
not only do the benefits come with the cost of lower programmability, but also the acceleration requires long development cycles and extensive expertise in hardware design. Moreover, conventionally, accelerators are integrated with the existing computing stack by profiling hot regions of code and offloading the computation to the accelerators. This approach is suboptimal since the stack is designed and optimized merely for CPUs, the sole processing platform up until very recently. To tackle these challenges, cross-stack, and algorithm-hardware co-designed solutions that rebuild the computing stack for the acceleration of machine learning. These solutions break the conventional abstractions of the computing stack by reworking the entire layers of the computing stack, which include programming language, compiler, system software, accelerator architecture, and circuit generator.

Approximate computing is another form of specialization, which brings forth an unconventional yet innovative computing paradigm that trades accuracy of computation for otherwise hard-to-achieve performance and efficiency. This new computing paradigm is built upon the property that emerging applications (e.g., sensor processing, translation, vision, and data analytics) are increasingly tolerant of imprecision. Leveraging this property, approximation techniques can provide orders of magnitude higher performance and efficiency gains, while maintaining the acceptable level of functionalities. However, these techniques are only pragmatic when they are easy to use for the programmers, and they produce acceptable output quality from the perspective of application users. To this end, my research efforts for approximation focus on improving the productivity and utility of approximation technologies by developing programming language and crowdsourcing-based software engineering solutions.

2. This assignment requires you to review several references on RISCV beginning with a summary transcript [2] of the Debate on Proprietary Vs Open-Source Instruction Sets at the 4th Workshop on Computer Architecture Research Directions, June 2015 sponsored by the ACM.

This Debate between Professor David Patterson (author of the textbook you are using) and Dave Christie of AMD highlights all the key technical and business arguments for and against an Open-Source ISA such as RISC V as of 2015 (the same year the RISC V Foundation was established). A Technical Report from EECS UC Berkeley highlights the technical reasons for Open ISAs [3] providing a more detailed discussion of the advantages offered by open-source ISAs

(1) Articulate your views on the topics debated in [2]. Justify your views.

1)

While I agree with Davie Christie of AMD that the strong ecosystem developed around the mainstream commercial ecosystem is a formidable hurdle for any open-source ISA, I believe that Professor David Patterson's views on Open-Source Instruction sets are meticulous for Microprocessor Industry.

I agree with Professor's views that ISAs do matter. They're the point where hardware and software collide. The cost of porting software to a new processor account for most of the cost of the chip. The fact that ISAs are proprietary is not an oversight. Patents on ISA oddities are held by corporations with successful ISAs, preventing others from adopting them without permission. You cannot design an ARM core with an ARM license; you can only utilize Arm's designs. Only roughly ten to fifteen large corporations have licenses that allow them to create bespoke ARM cores. By preventing many people from building ISA-compatible cores, licenses impede competition and innovation. Proprietary ISAs are linked to the success of a single business and perish with it.

Engineers may share their designs using open-source software, which leads to more innovation. Because engineers do not have to develop the cores themselves, the time to market and cost will be reduced. Because processors for the Internet of Things must cost less than a dollar, the cost is becoming a critical consideration. Because more individuals are looking at and debugging the designs, the number of faults is reduced.

The ISA is integrated into the hardware of a 20th-century design. A corporation like Intel creates the microprocessor in a 21st-century architecture. This microprocessor must be capable of handling anything. The main distinction between monolithic and modular ISAs is that software only runs based on the latter. Open-source ISAs may usher in the age of minimum modular ISAs for SoCs.
In the past, open ISA techniques were tried, and they didn't work. There was no business need for them when they were launched during the monolithic microprocessor era. Intel, for example, sought to force the Itanium architecture on everyone ten years ago. Only because there was a second source, AMD, which is uncommon for proprietary ISAs nowadays, did it not work.

While the debate is from 2016, RISC-V has made much progress through the years. The introduction of RISC-V fills a longtime void in the computing industry. It will allow users to take control of their destiny and establish an open framework to fuel global collaboration and innovation. In addition to traction in the corporate world, more than two dozen universities are on board. "It will eventually supplant x86 and ARM as the primary instruction set for microprocessors," says Michael Taylor, associate professor at the University of Washington.
(2) Review and summarize technical reasons for Open-Source ISAs in

[3]. 2)

Open-Source Instruction Set Architecture would lead to a free open market of processor designs which could result in:

- Innovation via Competition: When a market is saturated with competitors, companies must differentiate themselves and their products in meaningful ways. This practice naturally lends itself to rapid innovation. Historically, differentiation has been a key motivator in the expedient development of advanced technologies.

- Shared Open-Source Design: Creates efficiency in the design process as designers don't waste time in solving problems already solved by the community which saves time and leads to a shorter time to market. Teaches other people best practices. Designers can learn from work others have done, like creating accessible user interface controls, color combinations, and other elements. Helps design teams get better. Releasing work to the broader community helps designers get valuable feedback to improve their work. Also, the more people work on the same design, the less scope of error is present in the final product.

- Increasing Affordability: Reusing available material will reduce the cost of research and development. Typically, open source implies that you are not obligated to pay for the use of designs. There is no need to use procurement overhead to manage license renewals. It is highly beneficial to enterprises to save budget to utilize elsewhere.

Key requirements for Open-Source Instruction Set Architecture:

- SoCs have unique application-specific accelerators to boost efficiency and lower costs. A free, open ISA should have the following features to meet the demands of SoCs while keeping a stable software base:
  i) a tiny core set of instructions on which compilers and operating systems can rely.
  ii) standard but optional ISA extensions to help tailor the SoC to the application. iii) room for completely new opcodes to call application-specific accelerators.

- Encoding of a small instruction set. Given the cost sensitivity of IoTs and the accompanying demand for less memory, smaller code is desired.

- QP floating point, as well as SP and DP floating-point. Some WSC applications currently process such massive data files that they already rely on QP arithmetic software packages.
- 32-bit, 64-bit, and 128-bit addressing. Because of the small memory sizes of IoTs, 32-bit addressing will continue to be significant for decades, while 64-bit addressing is the de facto standard in anything larger. Although the WSC industry will not require 2(128) bytes, it is possible that within a decade, WSCs may require more than 2(64) bytes (16 exabytes) to handle all their solid-state nonvolatile storage needs. Because the one ISA error from which it is difficult to recover is address size, it is prudent to plan for larger addresses today.

Open ISAs have been tried before, but due to a lack of demand, they were never popular. IoTs' low cost and power, the desire for a WSC alternative to the 80x86, and the fact that cores make up a small but ubiquitous portion of all SoCs all combine to fill that gap. RISC-V is designed for SoCs, with a foundation that should never change due to the endurance of core RISC principles, a standard set of optional extensions that will expand slowly, and unique instructions per SoC that will never be repeated.