Senior Thesis in Computer Engineering
University of Illinois at Urbana-Champaign
Advisor: Rakesh Kumar

Procurability-Aware Design Space Exploration

Lin Chuan Lee

Submitted in part fulfilment of the requirements for the degree of
Bachelor of Science in Computer Engineering of the
University of Illinois at Urbana-Champaign, May 2023

Abstract

The unpredictable demand, complex supply chain, and high barrier of entry of the semiconductor industry make it prone to shortages. Consequently, creating designs with highly procurable, easily replaceable components has become an increasing priority for any businesses relying on semiconductors: from car makers and home appliance manufacturers to the many companies producing consumer and commercial electronics. For instance, while High Bandwidth Memory (HBM) memory offers high bandwidth, low power consumption, and a small form factor, it has lower procurability and higher cost compared to traditional off-chip DRAM DIMM due to the limited number of suppliers and the use of interposers.

In this work, we performed design space exploration (DSE) to find design points using DDR memory with various L3 capacities, number of memory channels, and memory frequency with reasonable performance and cost compared to a system using HBM memory. The DSE is performed through an analytical model that estimates the performance, area, power, and cost of design points. Using a roofline model, we found regions where the system performance is constrained by bandwidth between compute core and L3 cache, bandwidth between the L3 cache and main memory, and compute throughput. We found, for the same L3 capacity, the HBM system has the highest memory cost and total cost but the lowest package cost compared to DDR systems. We demonstrated that, across a range of application profiles with different bandwidth requirements, the HBM system has the highest system cost but outperforms DDR systems in terms of compute die area, package area, and compute die power in iso-performance comparisons. Finally, we found design points using DDR memory that offers the same performance as the reference HBM system at a lower cost, even when lifetime energy costs are considered.

Subject Keywords: architectural design space exploration (DSE); HBM memory; supply chain @@ please advise

Contents

Chapter 1

Introduction

Along with the COVID-19 pandemic, more than 169 countries around the world observed a chip shortage in 2020 [1]. The automotive industry, one of the many industries relying on semiconductors, reached an all-time low production volume in the U.S. since 1993 [2] and was expected to lose $210 billion in revenue in 2021 [3]. Shortages occur when there is a mismatch between demand and supply. The semiconductor industry has an unpredictable demand due to the rapid pace of technological innovation that requires different types of chips in terms of architecture, power, and form factor. It is equally difficult to predict what new technology will become dominant in the market. Production of chips involves a complex supply chain from upstream equipment suppliers like ASML to downstream companies doing integrated circuit (IC) assembly and testing. Each part of this interlocked supply chain can become the bottleneck [4]. With unpredictable demand and supply in the semiconductor industry, we are likely to see chip shortages in the future. Given the wide range of industries that depend on IC chips for their products, developing approaches to managing chip shortages is key.

In addition to stockpiling inventory, observing the supply chain in advance, and other marketing strategies to influence consumer behavior, companies using semiconductors can overcome supply chain disruptions by deploying engineering techniques to design products that rely less on difficult-to-procure components and use modular architectures that allow quick substitution of unavailable components [5]. In other words, given the state of the semiconductor industries, designing with procurability in mind should become a high priority for any companies relying on semiconductors. The procurability of a component depends on several factors, including the number of suppliers, the production volume, the cost of the component, and the cost of integrating the component with the existing system. The procurability of a design can be increased by using components that are readily available on the market from a wide selection of suppliers based on matured technology. Procurability-aware designs use standard interfaces and protocols and avoid reliance on custom parts.

The goal of this work is to consider procurability as a first-order metric in system architecture design by developing a design space exploration (DSE) framework that incorporates procurability. We assume a generic multi-core system with compute cores, cache (private L1 and L2 with shared L3), memory controllers, and I/O controllers. We constructed an analytical model of the system that estimates the performance, power, area, and cost of a given system configuration. Since the main memory bandwidth is a key factor in influencing the performance of modern computing systems, we consider different implementations of main memory and the corresponding trade-off in performance, cost, and procurability. Conventional main memory systems are implemented using off-chip DDR DRAM DIMMs. An emerging technology called High Bandwidth Memory (HBM) enables the on-chip integration of main memory [6]. The DRAM dies are vertically stacked and bonded to the compute die through a silicon interposer, resulting in a smaller form factor and lower power consumption. Furthermore, HBM is able to achieve much higher bandwidth than DDR with a much wider interface (1024 bits for HBM [7] versus 32 bits for GDDR [6]). However, HBM is less procurable because of a smaller number of suppliers and a higher integration cost due to the use of interposers.

Our goal is to apply our DSE framework to identify highly procurable design points that replace HBM memory in our reference system with DDR memory while maintaining comparable performance and cost. We compensate for the bandwidth difference between DDR and HBM memory by exploring different L3 capacities, number of memory channels, and memory frequencies.

Using parameters from modern systems, we observed regions where the system performance is bounded by compute core-L3 bandwidth, L3-main memory bandwidth, and compute throughput. For the same L3 capacity, we observed that the reference HBM system has a higher memory cost and total cost but a lower package cost compared to any DDR systems. Across a range of application profiles from compute to memory intensive, we demonstrated that the HBM system has the highest system cost but outperforms DDR systems in compute die area, package area, and compute die power for iso-performance comparisons. This means there exist design points where HBM memory can be substituted with DDR memory for the same performance at a lower system cost by incorporating a larger L3 cache, more memory channels, a higher memory frequency, or some combination of the three. This conclusion still holds even when the lifetime energy cost is considered.

1.1 Thesis Outline

This thesis has been organized into six chapters.

- In Chapter 2, we consider the state of the semiconductor industry that makes procurability an important design criterion.
- In Chapter 3, we describe the system that we want to model in our DSE.
- In Chapter 4, we discuss the formulation of the analytical model and describe how it estimates systems' performance, cost, area, and power.
- In Chapter 5, we discuss the set of design points of interest to be considered in our DSE.
- In Chapter 6, we describe the results from applying our analytical model to the set of design points.
- In Chapter 7, we summarize the main findings of this work.

Chapter 2

Background

2.1 Procurability-Aware Designs

The global chip shortage that began in 2020 impacted more than 169 industries [1], with the automobile industry alone expected to lose $210 billion in revenue in 2021 [3]. Figure 2.1 shows the U.S. automobile production trend from 1993 to 2022. The production volume reached an all-time low in April 2022. The global shortage was caused by a multitude of factors, including the COVID-19 pandemic's work-from-home economy boosting demand for consumer electronics [8], political tension between the U.S. and China resulting in restrictions on semiconductor imports and exports [9], proof-of-work cryptocurrency mining increasing demand for GPUs [10], and even natural phenomenons like drought impacting foundry production [11]. The semiconductor industry has unpredictable demand due to the rapid emergence of different applications that require different kinds of silicon chips. The supply is also unpredictable since each part of the complex supply chain can become the bottleneck. The industry's low-margin nature also means foundries are often operating at close to max capacity [4] without a redundancy buffer. The global chip shortage of 2020 demonstrated that the semiconductor industry is susceptible to shortages whenever there is a mismatch between demand and supply.

As the reach of computing continues to expand, companies from a wide range of industries depend on semiconductors for their products. Figure 2.2 shows the main industries that rely

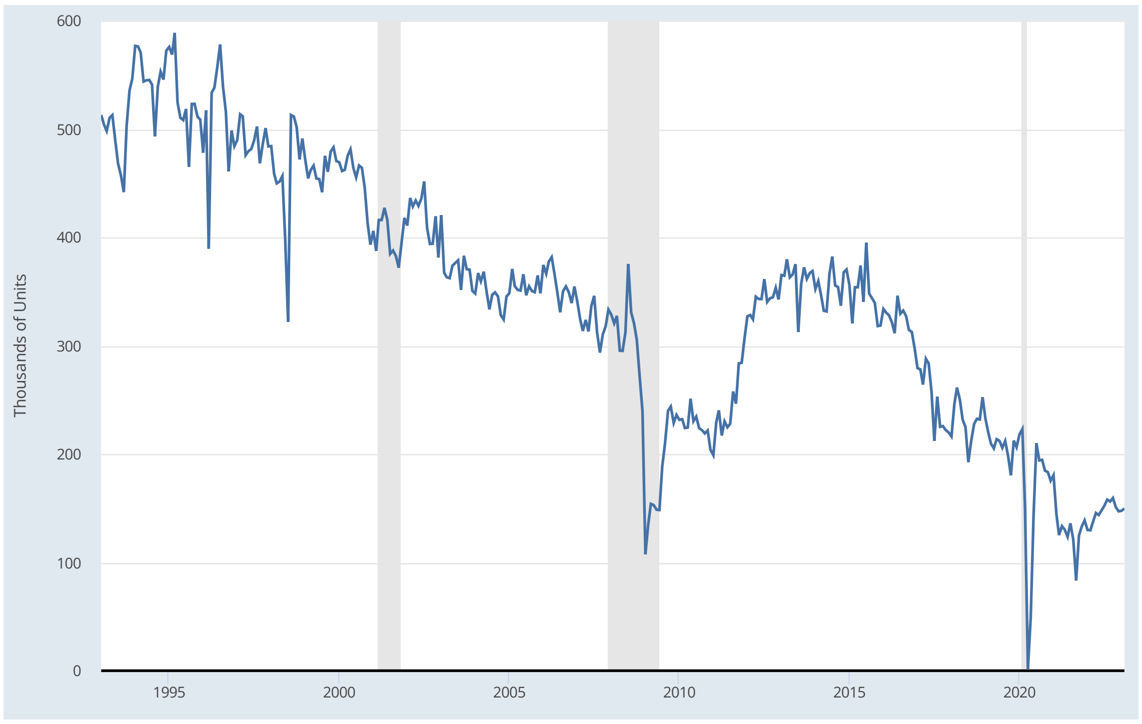

Figure 2.1: U.S. automobile production volume trend from 1993 to 2022. Shaded areas indicate U.S. recessions.

Source: Adapted from [2].

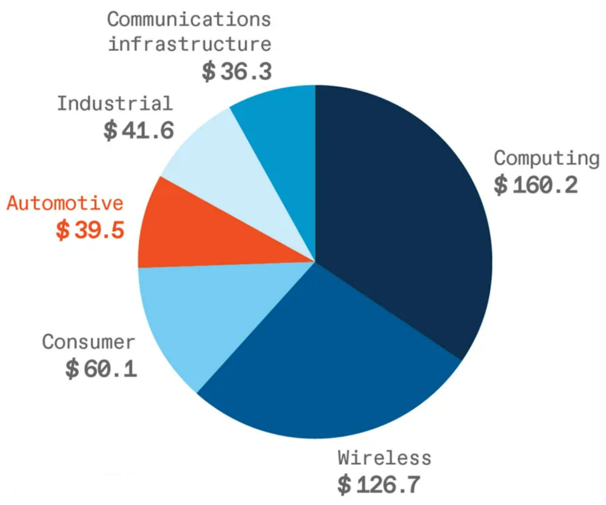

Figure 2.2: Semiconductor chips demand by revenue in billions of USD, according to market research firm International Data Corporation (IDC).

Source: Adapted from [12].

on semiconductor chips by revenue. For these businesses, resilience to disruption in the semiconductor supply chain is key. One of the ways to prepare for these shortages is to consider procurability in product architectures. Instead of using components with a small number of suppliers, companies design products with modular and flexible architectures that allow the substitution of components when a subset of components becomes unavailable [5].

2.2 HBM Memory

One such example of hard-to-procure components is HBM memory. DRAM dies in HBM are vertically stacked and connected by through-silicon via (TSV) and micro bumps. The HBM DRAM stack is then integrated into the same package as the compute die through a silicon interposer, resulting in a smaller form factor compared to off-chip DRAM DIMMs. HBM offers much higher bandwidth compared to DDR memory by employing a much wider interface (1024 bits for HBM [7] versus 32 bits for GDDR [6]). The lower frequency and shorter interconnect distance through the interposer mean HBM also has lower power consumption. However, HBM memory has low procurability due to a limited number of suppliers (Micron Technology, Samsung, and SK Hynix are the only main manufacturers [13]), which also contributes to its high cost. HBMs are also expensive due to the use of silicon interposers (silicon fabrication is more costly than PCB production). Thus, in this work, we perform a design space exploration to search for procurability-aware designs by substituting HBM with DDR memory. We attempt to compensate for the bandwidth difference by a combination of larger L3 capacities, more memory channels, and high memory frequencies.

Chapter 3

View of the System to Be Modeled

In this chapter, we discuss the view of the system to be modeled in our design space exploration.

3.1 System Architecture

In this section, we describe the architectural organization of the system. The organization of the system is shown in Figure 3.1. We assume a generic multi-core system with L1, L2, and L3 cache. Each *compute core* in the system, with its own private L1 and L2 cache, is

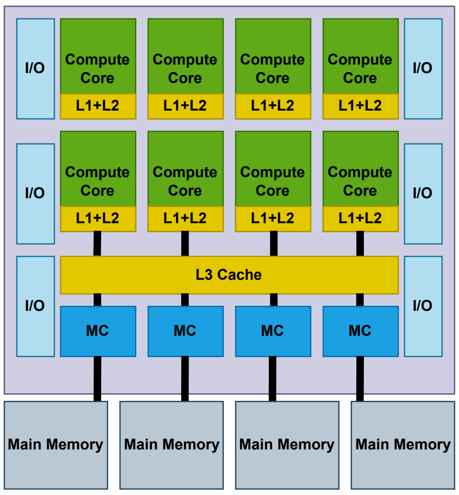

Figure 3.1: Architectural organization of the system to be modeled with compute cores (including L1 and L2 cache), L3 cache, memory controllers, and I/O controllers.

assumed to be architecturally identical with the same IPC. L1 and L2 caches are assumed to be exclusive of one another. All compute cores share the last level cache (L3). The multi-core system communicates with the main memory through a number of memory controllers. Each memory controller adds an independent memory channel with its own command and data buses. A number of I/O controllers interface the system with GPUs, external accelerators, network cards, and human interface devices. Thus, the four components of the modeled system are compute cores (including L1 and L2 cache), L3 cache, memory controllers, and I/O controllers. We use the term *memory subsystem* to refer to the memory hierarchy consisting of L3 and the main memory. We also assume that all memory accesses originating from the compute cores are serviced by the L3 cache; only L3 misses are serviced by the main memory.

3.2 System Physical Organization

In this section, we describe the physical organization of the system. Since systems using HBM as main memory require interposers, the physical structure will be different. Figures 3.2 and 3.3 show the cross-section of the physical organization of systems using DDR and HBM memory, respectively. *Compute die* refers to the die containing the compute cores (including L1 and L2 cache), L3 cache, memory controllers, and I/O controller (i.e. the four components of the modeled system). Systems using conventional DDR DRAM have the compute die bonded directly onto the package. Since the DRAM DIMMs are off-chip, the DDR interface signals are routed through the package and across the PCB. In contrast, systems using HBM main memory have on-chip DRAM dies. These HBM DRAM dies and base logic die are stacked vertically and connected by through-silicon vias and micro bumps. The HBM stack is then integrated with the compute die via a silicon interposer. Since the DRAM dies are on-chip, the package requires additional bumps for power delivery. However, the package lacks bumps for the memory interface since HBM interface signals are routed through the interposer within the package. Another feature of the HBM system is that the compute and memory dies can be bonded to the interposers at a much lower bump pitch (50 μm versus 150 μm) compared to conventional die-to-package bonding. However, the current rating of each bump also scales linearly with bump pitch squared (58 mA versus 520 mA).

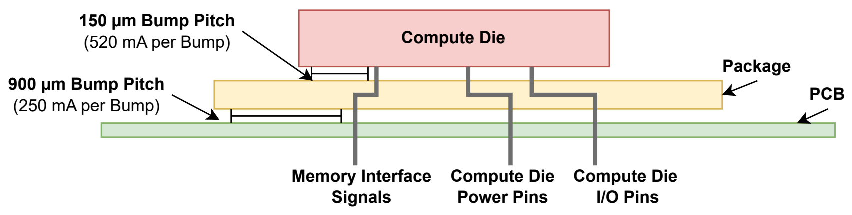

Figure 3.2: Cross-section of systems using DDR memory is shown, with bump pitch and bump current rating labeled.

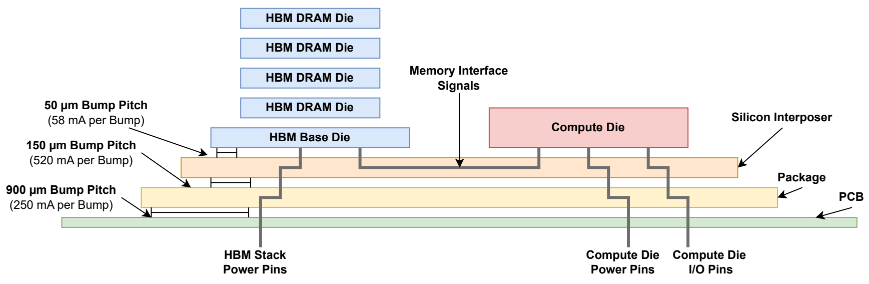

Figure 3.3: Cross-section of systems using HBM memory is shown, with bump pitch and bump current rating labeled.

Chapter 4

Analytical Model of the System under Study

In this chapter, we describe the formulation of our analytical model that we used to describe the system discussed in Chapter 3. The analytical model consists of three main parts. First, for each design point, the model estimates the performance via a roofline model that considers L3 cache hit rate, L3 bandwidth, main memory bandwidth, and compute throughput. Second, the model estimates the power and area of each design point to filter out unrealistic designs. Third, for each design point, the model estimates the cost of the compute die, memory, interposer (for HBM systems), and package.

4.1 Performance Model

Although there are many factors influencing the overall performance of a given system, we assume the system's performance is only impacted by the compute throughput of the compute cores, the arithmetic intensity, and the bandwidth of the memory subsystem. Thus, we need to model all of three factors.

To model the compute throughput of the compute cores, we made a simplifying assumption that each compute core is independent and does not interfere with one another. This allows us to estimate the compute throughput of the multi-core system as the sum of the compute throughput of each compute core. We further assume all compute cores have the same microarchitecture, thus the same cycles per instruction, and operate at the same frequency. Under these assumptions, the compute throughput of the system is simply given by

$$\pi = \mathit{CPI} \cdot f_{\text{core}} \cdot N_{\text{core}} \tag{4.1}$$

where CPI is the cycles per instruction, f_{core} is the frequency of the compute cores, and N_{core} is the number of compute cores.

To model the bandwidth of the memory subsystem consisting of L3 cache and main memory, we need to consider the bandwidth between the compute cores and L3 and the bandwidth between L3 and the main memory. We assume that all memory accesses from the compute cores are serviced by L3, and only L3 misses are serviced by the main memory. Since, in modern computing systems, the compute cores can issue memory access instructions much faster than L3 can process them, we assume that the bandwidth between the compute cores and L3 is determined by the ability of the L3 cache to handle requests from the compute cores. Thus, the compute core-L3 bandwidth is given as $N_{\text{L3}} \cdot B_{\text{L3}}$ where N_{L3} is the number of L3 slices and B_{L3} is the bandwidth per L3 slice. We vary L3 capacity at 2 MB step size in our DSE, so a slice of L3 is 2 MB (see Chapter 5). The compute core-L3 bandwidth scales linearly with L3 capacity (number of L3 slices) since a larger L3 cache has more read and write ports that can operate in parallel. For the same reason, we assume the bandwidth between L3 and the main memory is determined by the ability of the main memory to process read and write requests from the L3 cache. Thus, the L3-main memory bandwidth is given as $N_{\text{MC}} \cdot B_{\text{MC}}$ where N_{MC} is the number of memory controllers and B_{MC} is the bandwidth per memory controller. Each memory controller adds an additional memory channel that operates in parallel with its own command and data interface, increasing the bandwidth. However, the bandwidth of the memory subsystem is not simply the minimum of $N_{\text{L3}} \cdot B_{\text{L3}}$ and $N_{\text{MC}} \cdot B_{\text{MC}}$. Even if the compute core-L3 bandwidth is higher than the L3-main memory bandwidth, the bandwidth of the memory subsystem can still be bounded by the former if the L3 hit rate is high, which means the main memory only has to process a small fraction of requests compared to L3 cache. A higher L3 hit rate implies a better utilization of the available L3-main memory bandwidth. Hence, the memory subsystem bandwidth is given as

$$B_{\text{msys}} = \min\left(N_{\text{L3}} \cdot B_{\text{L3}},\ \frac{N_{\text{MC}} \cdot B_{\text{MC}}}{\text{L3 Miss Rate}}\right) \tag{4.2}$$

where N_{L3} is the number of L3 slices, B_{L3} is the bandwidth per L3 slice, N_{MC} is the number of memory controllers, and B_{MC} is the bandwidth per memory controller. The first term is the bandwidth between compute cores and L3, as explained, and the second term is the *effective L3-main memory bandwidth* (i.e. the effective bandwidth used when determining the memory subsystem bandwidth).

To compute the L3 hit rate, we assume the system is in a steady state and ignore the compulsory misses during cache warm-up. We also assume the L3 cache is fully associative for the purpose of estimating the hit rate so we can disregard the effect of conflict misses. Then, L3 hit rate depends only on the number of capacity misses, which depends on the relative sizes of the application working set and the L3 cache. If the working set size exceeds the L3 capacity, parts of the working set will be constantly evicted, and the hit rate will decrease due to capacity misses. With these assumptions, we model the L3 hit rate as scaling linearly with the ratio of the L3 capacity to the working set size until the entire working set fits within L3, at which point we assume a nominal hit rate of 90%.

With the compute throughput and bandwidth of the memory subsystem given by Equations 4.1 and 4.2, respectively, the only other factor is the arithmetic intensity. The arithmetic intensity must take into consideration the filtering effect of the L1 and L2 cache, which is part of the compute cores, since load and store requests are first serviced by the L1 and L2 cache before they are sent to the memory subsystem. To estimate the extent of this filtering effect, we again assume L1 and L2 cache are fully-associative and ignore cold start misses. Then, the degree of filtering depends on the size of the working set and the L1 and L2 capacity. If L1 and

L2 cache are non-existent, then the arithmetic intensity seen by the memory subsystem, I_{msys}, should be equal to the arithmetic intensity that is inherent to the application or kernel, I_{app}. If the L1 and L2 cache are half the size of the working set, then, on average, only 50% of the memory references to working set data will miss in the L1 and L2 cache and proceed to the memory subsystem. Consequently, I_{msys} should be twice I_{app}. Finally, as L1 and L2 capacity approaches the working set size, I_{msys} should approach infinity since the L1 and L2 cache are nearly containing everything from the working set. Thus, the relationship between I_{msys} and I_{app} is given as

$$I_{\text{msys}} = \frac{Cap_{\text{workset}}}{Cap_{\text{workset}} - (Cap_{\text{L1}} + Cap_{\text{L2}})} \cdot I_{\text{app}} \tag{4.3}$$

The L1 and L2 capacity are summed together in Equation 4.3 since L1 and L2 cache are assumed to be exclusive of one another.

Finally, the performance of the overall multi-core system is given as

$$Perf = \min(\pi,\ B_{\text{msys}} \cdot I_{\text{msys}}) \tag{4.4}$$

where the first term is the compute throughput of the compute cores, and the second term is the *effective memory subsystem bandwidth* (i.e. the effective bandwidth used when determining the overall system performance). It is the FLOP rate that the memory subsystem can support, given the arithmetic intensity inherent to the application or kernel and the filtering effect of the L1 and L2 cache.

This concludes our discussion of the performance model.

4.2 Power Model

To ensure all evaluated design points have reasonable power overhead, we developed the power model. The power model consists of two parts. The first part computes the amount of

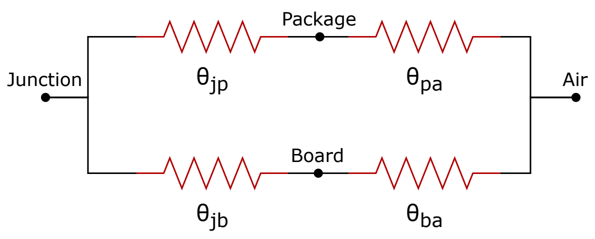

Figure 4.1: Thermal resistance network for the system.

heat that can be dissipated by the system's packaging. The second part computes the power consumption of all of the system's components. Then, we compare the two values to ensure the power consumption of the system does not exceed its power envelope.

To estimate the amount of heat that the system's packaging can dissipate, we assume that the thermal energy generated by the system at the junction can be dissipated through two parallel paths [14]. The first path is from the junction to the package and to the heat sink. The second path is through the back of the PCB. The configuration of the thermal resistance network is shown in Figure 4.1, and the analytical form is given as

$$\theta_{\text{ja}} = \frac{(\theta_{\text{jp}} + \theta_{\text{pa}})(\theta_{\text{jb}} + \theta_{\text{ba}})}{\theta_{\text{jp}} + \theta_{\text{pa}} + \theta_{\text{jb}} + \theta_{\text{ba}}} \tag{4.5}$$

where θ_{jp}, θ_{pa}, θ_{jb}, and θ_{ba} denote the thermal resistance from junction to package, package to air, junction to PCB, and PCB to air, respectively. Since the rate of thermal energy transfer depends on the temperature differential, we assume an ambient operating temperature of 25 $°C$ and a maximum junction temperature of 110 $°C$. Then, the maximum thermal envelope of the system is given as

$$P_{\text{max}} = \frac{T_{\text{j, max}} - T_{\text{ambient}}}{\theta_{\text{ja}}} \tag{4.6}$$

Then, we compute the power consumption of the system to ensure it does not exceed P_{max}, which involves computing the power of compute cores, L3 cache, memory controllers, and I/O controllers separately. We made a simplifying assumption that the entire compute die is supplied at the same voltage level, with the exception of the memory controller physical layer

(PHY).

The power of the compute cores consists of dynamic, leakage, and short-circuit power. As a simplifying assumption, we assume the activity factor of the transistors is high enough such that dynamic power accounts for most of the power consumption, and we can disregard leakage and short-circuit power. Dynamic power is computed as $C \cdot V^2 \cdot f$ where C, V, and f are the capacitance, voltage, and frequency of the compute cores, respectively. Since different operating voltage levels are associated with different frequency ranges at which the system can operate reliably, we scale linearly the voltage of the compute cores with the operating frequency. In other words, a reduction in operating frequency gives a cubic reduction in dynamic power.

The memory controller consists of the controller's digital logic and the PHY, which interfaces the controller with the DRAM device. Since the DRAM device's frequency is much lower than the compute core frequency, we assume that the memory controller, including the digital logic and the PHY, operates at the same frequency as the DRAM device. The PHY voltage is scaled linearly with the memory controller frequency for the same reason as above. For the power of the PHY, we assume that most of the power is used to transmit signals from the compute die to the DRAM device and is given by

$$P_{\text{PHY}} = E_{\text{bit}} \cdot f_{\text{MC}} \cdot \#Wire_{\text{MC}} \cdot \left(\frac{V_{\text{MC}}}{V_{\text{MC, nominal}}} \right)^2 \tag{4.7}$$

where E_{bit} is the energy required to signal a bit from the compute die to the DRAM device, f_{MC} is the frequency of the memory controller and DRAM device, $\#Wire_{\text{MC}}$ is the number of wires per memory controller, and V_{MC} is the PHY voltage. In other words, the PHY power depends on the energy cost per bit (which will vary significantly for on-chip versus off-chip DRAM), frequency of transmission, the number of wires that need to be driven, and the PHY voltage squared. The power of the digital logic also depends linearly on the memory controller frequency. It is given by Equation 4.8. Thus, the total power of each memory controller is given by the sum of P_{PHY} and P_{logic}.

$$P_{\text{logic}} = \frac{f_{\text{MC}}}{f_{\text{MC, nominal}}} \cdot P_{\text{controller, nominal}} \tag{4.8}$$

For the L3 power, we made the simplifying assumption that the number of cache read and write hits scale linearly with cache capacity since a larger proportion of the working set will be cached. We also assume tag comparison for cache misses consumes a negligible amount of power. These two assumptions mean the L3 power can be modeled simply as a linear function of L3 capacity. The I/O controller power is also a linear function of the number of controllers. For systems with HBM memory, the power consumption of the DRAM stack also contributes to system power since the DRAM stack is integrated within the package. The final system power consumption is the dot product between the vector of component counts and the vector of component powers.

4.3 Area Model

To ensure all evaluated design points have reasonable area overhead, we developed the area models. The area model estimates the total area of all of the system's components, which we compare against lower and upper bounds. The lower bound is set to ensure that (i) there is enough space on the back of the compute die to accommodate all bumps required for signaling and power delivery and (ii) there is enough perimeter length on the compute die for wire fan-out. The upper bound is set as part of the experimental setup (see Chapter 5) to filter out designs with unrealistic areas.

The area of the system is simply the sum of compute core, L3 cache, memory controller, and I/O controller areas. To increase the compute core frequency beyond a certain point, microarchitectural and low-level synthesis adjustments are required to meet timing closure in addition to increasing the operating voltage. Specifically, these adjustments that increase compute core area impact logic versus memory differently. A 5% increase in compute core frequency beyond the baseline results in a 10% increase in compute core logic area and a 2% increase in compute core area dedicated to L1 and L2 cache. For L3 cache, memory controllers, and I/O controllers, we assume that the area is simply a linear function of the number of components. This is true for L3 cache because, in reality, caches are not implemented as a monolithic structure but as a set of smaller SRAM arrays.

Next, we compute the lower bound on the compute die area due to the bump constraint. Figures 3.2 and 3.3 show that both DDR and HBM systems have three types of bumps that are fitted on the back of the compute die, which are (i) memory interface bumps, (ii) I/O interface bumps, and (iii) power bumps for supplying power to the compute die. The compute die area must be large enough to accommodate these bumps. The bump pitch is also different for bonding to packages (DDR systems) versus interposers (HBM systems). The number of memory and I/O bumps is simply a linear function of the number of memory and I/O controllers, respectively, since each controller adds an additional set of wires. The number of power bumps depends on how much power the compute die consumes. The amount of power each pair of bumps (VCC and GND) can supply depends on the compute die voltage and the amount of current the bump is rated for. We assume the current rating per bump scales with bump pitch squared. Thus, the number of power bumps is given as

$$\#Bump_{\text{power}} = \frac{P_{\text{die}}}{V_{\text{die}} \cdot I_{\text{bump}}} \cdot 2 \tag{4.9}$$

where P_{die} is the compute die power, V_{die} is the compute die voltage, and I_{bump} is the bump current rating. Since each of the three types of bumps requires $(\text{Bump Pitch})^2$ area, the minimum compute die area is given by

$$A_{\text{die}} \geq (\text{Bump Pitch})^2 \cdot \left(\#Bump_{\text{power}} + \#Bump_{\text{MC}} \cdot N_{\text{MC}} + \#Bump_{\text{I/O}} \cdot N_{\text{I/O}}\right) \tag{4.10}$$

where $\#Bump_{\text{power}}$ is the number of bumps for compute die power delivery, $\#Bump_{\text{MC}}$ is the number of bumps per memory controller, N_{MC} is the number of memory controllers, $\#Bump_{\text{I/O}}$ is the number of bumps per I/O controller, and $N_{\text{I/O}}$ is the number of I/O controllers. If the die area is too small to fit all the bumps, we allocate as much dead space on the compute die as required as long as the upper bound is not violated. A design point with a large amount of dead space on the compute die will be automatically penalized by the cost model since the compute die will have poor yield and few compute dies can be cut from the wafer (see Section 4.4). For systems using DDR memory, we assume a compute die to package bump pitch of 150 μm with a current rating of 520 mA (see Figure 3.2). For systems using HBM memory, we assume a compute die to interposer bump pitch of 50 μm with a current rating of 58 mA (see Figure 3.3).

Next, we compute the lower bound on the compute die area due to the wire fan-out constraint. There must be enough perimeter length on the compute die for wire fan-out. To convert the compute die area into perimeter length, we assume the compute die has a 3:2 aspect ratio. The length of the perimeter for a given compute die area is $10 \cdot \sqrt{\frac{A_{\text{die}}}{6}}$. Along the perimeter of the compute die, wires are spaced link pitch apart for a number of horizontal layers depending on the PCB used. Thus, the maximum number of wires that can be routed across the compute die perimeter is given by

$$\#Wire_{\text{max}} = \left(10 \cdot \sqrt{\frac{A_{\text{die}}}{6}}\right) \cdot \frac{\text{PCB Layer}}{\text{Link Pitch}} \tag{4.11}$$

Since power delivery wires carrying high currents can cause interference and degrade signal integrity, a minimal routing distance is desired for these wires. Thus, we assume that power wires extend vertically downward from the compute die through the package or interposer without fanning out, which means there are only two types of wires that cross the compute die perimeter: (i) memory interface wires and (ii) I/O interface wires. Therefore, we enforce that

$$\#Wire_{\text{max}} \geq \#Wire_{\text{MC}} \cdot N_{\text{MC}} + \#Wire_{\text{I/O}} \cdot N_{\text{I/O}} \tag{4.12}$$

where $\#Wire_{\text{MC}}$ is the number of wires per memory controller, N_{MC} is the number of memory controllers, $\#Wire_{\text{I/O}}$ is the number of wires per I/O controller, and $N_{\text{I/O}}$ is the number of I/O controllers.

Both the bump constraint and wire fan-out constraint impose a lower bound on the compute die area. The stricter of the two will be the effective lower bound.

4.4 Cost Model

To meaningfully perform DSE, we must model the cost associated with each design point. The total cost is the sum of the cost of compute die, main memory, interposer (for HBM systems), and package. We will estimate each of these costs separately.

To estimate the cost of the compute die that includes compute cores, L3 cache, memory controllers, and I/O controllers, we must estimate the number of functional dies that can be harvested from a wafer, which depends on the yield and the total number of dies that can be cut from a wafer (DPW). To model the yield accurately, we need to take into consideration the fact that logic versus memory regions impact yield differently. Memory structures, such as L1, L2, and L3 cache, are more regular architecturally than logic structures like arithmetic logic units, register stacks, and branch predictors, which means redundancy techniques can be deployed more effectively on memory circuits [15]. Defective submodules can be disabled and substituted with redundant, functional ones. We then made a simplifying assumption that only the logic structures will impact yield. We also assume that all components in the compute die except for L1, L2, and L3 cache have no memory structures. For the cache hierarchy, we use CACTI [16] to estimate the percentage area that is dedicated to SRAM cells (memory) versus peripheral circuitry (logic). See Appendix A for the cache organization parameters that were input to CACTI. This enables us to accurately define the portion of the overall compute die area that will impact yield. Then, the yield is given by the commonly used negative binomial model

$$Y_{\text{die}} = \left(1 + \frac{A_{\text{die, yield}} \cdot D_0}{\alpha}\right)^{-\alpha} \tag{4.13}$$

where $A_{\text{die, yield}}$ is the compute die area impacting yield, D_0 is the defect density, and α is the clustering factor [17]. Then, we estimate the DPW with the following geometry expression

$$DPW = d \cdot \pi \left(\frac{d}{4 \cdot A_{\text{die}}} - \frac{1}{\sqrt{2 \cdot A_{\text{die}}}} \right) \tag{4.14}$$

where d is the wafer diameter and A_{die} is the compute die area [18]. Finally, the number of functional dies that can be harvested from a wafer is simply $DPW \cdot Y_{\text{die}}$. Given the wafer cost for a specific process node, we now have an estimate of the cost per compute die.

The cost of interposers, for systems using HBM memory, is similarly computed using Equations 4.13 and 4.14 with compute die area replaced with interposer area. Since interposers are less complex than compute dies, interposers have a lower defect density. We estimate the interposer area as the sum of the compute die area and the HBM DRAM stack area. We assume there is one HBM DRAM stack for each memory controller since each memory controller adds an additional memory channel. However, we also ensure that the interposer area estimated this way is large enough to fit all the die-to-interposer and interposer-to-package bumps. We also include the cost of assembling the compute die, HBM DRAM stack, and interposer.

For the package cost, we first assume that systems using HBM and DDR use the same type of package (i.e. they both have the same die/interposer-to-package and package-to-PCB bump pitch, as shown in Figures 3.2 and 3.3). Second, we assume the package cost is a linear function of the package area. Third, we estimate the size of the package as the amount of area required to fit all the package-to-PCB bumps. There are three types of bumps underneath the package: (i) bumps for power delivery, (ii) bumps for memory interface, and (iii) bumps for I/O interface. The area taken by these bumps is estimated in a way similar to Equations 4.9 and 4.10. The package area for DDR system is given as

$$A_{\text{pkg, DDR}} = (\text{Bump Pitch})^2 \cdot \left(\left[\frac{P_{\text{die}}}{V_{\text{die}} \cdot I_{\text{bump}}} \cdot 2 \right] + \#Bump_{\text{MC}} \cdot N_{\text{MC}} + \#Bump_{\text{I/O}} \cdot N_{\text{I/O}} \right) \tag{4.15}$$

where the term inside the square brackets is the number of bumps for power delivery. The package area for HBM system is given as

$$A_{\text{pkg, HBM}} = (\text{Bump Pitch})^2 \cdot \left(\left[\frac{P_{\text{die}} + (P_{\text{DRAM}} \cdot N_{\text{MC}})}{V_{\text{die}} \cdot I_{\text{bump}}} \cdot 2 \right] + \#Bump_{\text{I/O}} \cdot N_{\text{I/O}} \right) \tag{4.16}$$

where P_{DRAM} is the power consumption per HBM DRAM stack. For HBM systems, the memory interface wires connecting the HBM memory controller on the compute die with the HBM DRAM stack are routed within the interposer without exiting the package. This means HBM systems do not need package-to-PCB bumps for memory interface signals. Thus, Equation 4.16 lacks the $\#Bump_{\text{MC}} \cdot N_{\text{MC}}$ term. Another consequence of having on-chip DRAM dies for HBM systems is that the power of the DRAM dies needs to be supplied through the same package. Thus, in Equation 4.16, the total power supplied through power bumps is $P_{\text{die}} + (P_{\text{DRAM}} \cdot N_{\text{MC}})$ rather than simply P_{die} as in Equation 4.15.

We estimate the cost of main memory from commercially available products [19, 20, 21, 22]. For different types of main memory, prices are sampled at the same capacity of 16 GB per channel.

Now that we have estimated the cost of compute die, main memory, interposer (for HBM systems), and package individually, we simply have to sum them together for the total cost.

Chapter 5

Experimental Setup

With the performance, power, area, and cost model described in Chapter 4, the next step is to set up the experiment to perform the design space exploration (DSE).

5.1 DSE Framework

The goal of the DSE is to explore design points with high procurability by replacing DDR with HBM main memory. As a way to compensate for the lower bandwidth of DDR memory, we search for designs with different L3 capacities, the number of memory channels, and memory frequencies. The DSE is done with a search-based framework, as shown in Figure 5.1. We start with an initial set of design points of interest outlined in Section 5.2. For each of these design points, we apply the performance and cost model to compute the attainable performance and total cost, respectively. Then, we apply the power and area model on every design point to filter out designs with unreasonable power consumption and area. We enforce a power limit of 500 W and an area limit of 1000 mm^2 (power limit enforced through θ_{jp}, θ_{pa}, θ_{jb}, θ_{ba}, $T_{\text{j, max}}$, and T_{ambient}). We set generous upper bounds for power and area because inefficient designs with high power or area will be naturally penalized by the cost model. We are then left with a set of final design points, for each of which we have the performance, cost, and all other model variable values. While the framework we developed supports searching for the design point with

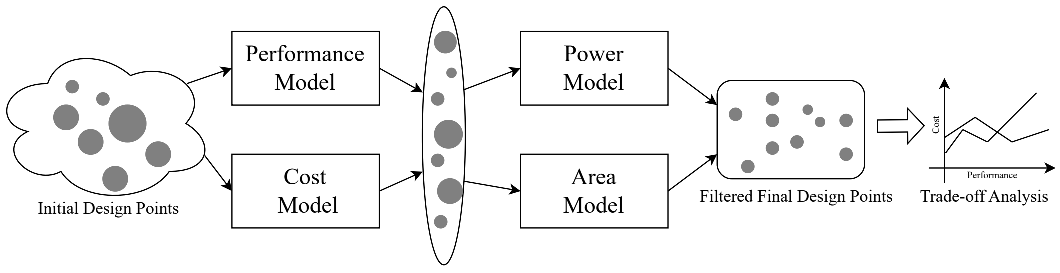

Figure 5.1: Framework for design space exploration.

max performance, minimum area, or minimum power, we retained all feasible design points for plotting and analysis.

5.2 Initial Design Space

As explained in Section 3.1, the components of the modeled system are compute cores (including L1 and L3 cache), L3 cache, memory controllers, and I/O controllers. Since our design space exploration is focused on exploring the trade-off between using DDR versus HBM as main memory, we keep the configuration of compute cores and I/O controllers constant. The independent variables of the experiment are the type of memory (DDR versus HBM), L3 cache capacity, number of memory channels, and memory frequency.

The compute core parameters are based on the Intel Xeon Platinum 8380 CPU [23]. Since we are interested in the performance and cost tradeoff between HBM and DDR memory, a high compute throughput CPU will ensure most design points will not be compute-bound. For the area of L1, L2, and L3 cache, we assume an L3 cache density of 0.5 MB/mm^2 with L2 and L1 cache being 1.5x and 4x less dense to account for the larger peripheral circuitry. We verified the area of the L1, L2, and L3 cache using CACTI [16] with cache organization parameters listed in Appendix A.

For L3 cache capacity, we consider capacity from 2 MB to 200 MB with a step size of 2 MB (each L3 slice is 2 MB). For the number of memory channels and memory frequency, we consider a number of DDR and HBM memory configurations (see Table 5.1).

Table 5.1: Memory Configurations Considered for DSE

Memory Type	**Frequency (MHz)**	**Channel Counts**
DDR4	2400, 3200	4, 6
DDR5	4800, 5600	4, 6
HBM2	1000	4

Table 5.2: Application Profiles Considered for DSE

Arithmetic Intensity (FLOPs/byte)	0.125, 0.25, 0.5, 1
Working Set Size (MB)	25, 50, 100, 150

Then, the set of initial design points is the full factorial combination of the ranges of each independent variable (i.e. all possible combinations of memory types, L3 cache capacities, memory configurations listed in Table 5.1). We apply the DSE framework shown in Figure 5.1 on this set of initial design points.

Finally, we repeat the experiment for different application profiles listed in Table 5.2 (i.e. different combinations of arithmetic intensities and working set sizes). Arithmetic intensity determines the regions in which the system is compute bound versus bandwidth bound. Working set size determines (i) the degree to which memory accesses from the compute cores are filtered by L1 and L2 cache before being serviced by the memory subsystem and (ii) the hit rate of the L3 cache in the memory subsystem.

5.3 Methodology

The analytical model described in Chapter 4 is implemented in Python with model input parameters stored in JSON files. CACTI is used for the estimation of cache area and the percentage of cache area dedicated to SRAM cells versus peripheral circuitry. Wafer costs for different technology nodes are based on TSMC's pricing for 300 mm wafers [24]. Through command line arguments, the framework supports the specification of any performance, power, and area constraints as well as objectives. For the design space exploration over all design points for all application profiles, the runtime is roughly around one minute.

Chapter 6

Results

In this chapter, we describe the results of applying our analytical model to the set of design points. We start by examining the attainable performance of design points. Then, we incorporate the cost model to identify design points with reasonable performance and cost.

6.1 Compute Bound versus Bandwidth Bound

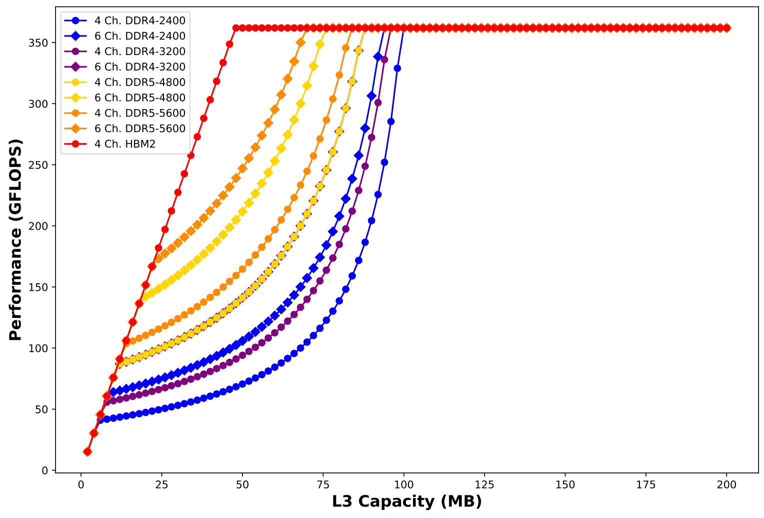

Figure 6.1: Line curve showing overall system performance versus L3 capacity. We show the curves for different memory types, frequencies, and channel counts. The plot is drawn for application arithmetic intensity of 0.5 FLOPs/byte and working set size of 100 MB.

Figure 6.1 shows the system performance of design points as a function of L3 capacity. A curve is drawn for a different combination of memory frequency and channel count. For the same L3 capacity, systems with higher memory frequency or more memory channels have better performance, as expected. Systems with HBM2 memory outperform, or have identical performance, compared to any DDR systems for any L3 capacity. To identify regions where the system performance is bounded by compute throughput, compute core-L3 bandwidth, or effective L3-main memory bandwidth, we have to examine the relationship between each of these quantities.

Figure 6.2 shows, as a function of L3 capacity, the bandwidth between compute core and L3 and the effective bandwidth between L3 and main memory for 4 channels of DDR5-4800 memory. The compute core-L3 bandwidth increases linearly as L3 capacity increases since adding additional slices of L3 cache increases the number of R/W ports. The effective L3-main memory bandwidth increases super-linearly because as L3 capacity increases for fixed working set size, the L3 hit rate increases, and the effective L3-main memory bandwidth has a $\frac{1}{1-x}$ dependence on the L3 hit rate (see Equation 4.2). However, the effective L3-main memory bandwidth stops increasing for L3 capacity larger than 100 MB because at that point the entire working set fits into the L3 cache.

Figure 6.3 shows, as a function of L3 capacity, the effective memory subsystem bandwidth and compute throughput also for 4 channels of DDR5-4800 memory (again, *memory subsystem* refer to the memory hierarchy consisting of L3 and the main memory). As shown in Equations 4.2 and 4.4, the effective memory subsystem bandwidth curve is the minimum of the two curves in Figure 6.2 multiplied by the arithmetic intensity. The compute throughput curve is constant since compute core parameters are not varied during the DSE. The minimum of the effective memory subsystem bandwidth and compute throughput curves gives the system performance curve for 4 channels of DDR5-4800 memory that was shown in Figure 6.1.

Based on these observations, we see that the system with 4 channels of DDR5-4800 memory is bounded by compute core-L3 bandwidth for L3 capacities from 2 MB to 12 MB. As we continue to add more slices of L3, the system performance becomes bounded by the effective

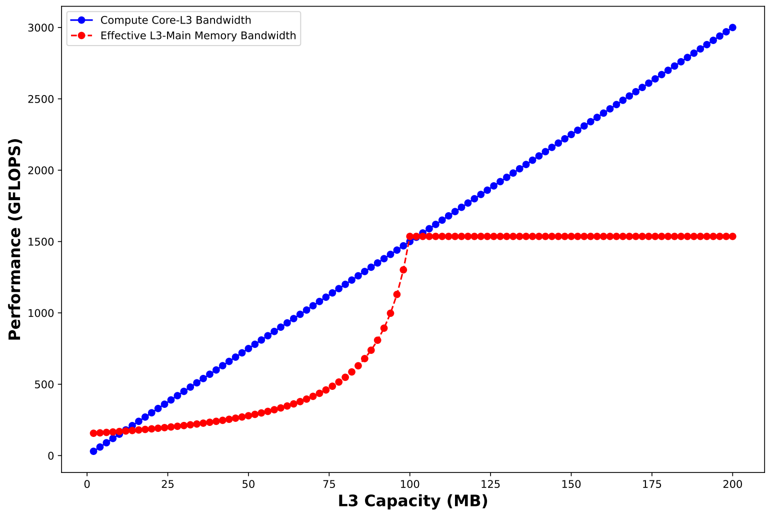

Figure 6.2: Line curve showing compute core-L3 bandwidth and effective L3-main memory bandwidth versus L3 capacity for 4 channels of DDR5-4800 memory. The plot is drawn for application arithmetic intensity of 0.5 FLOPs/byte and working set size of 100 MB.

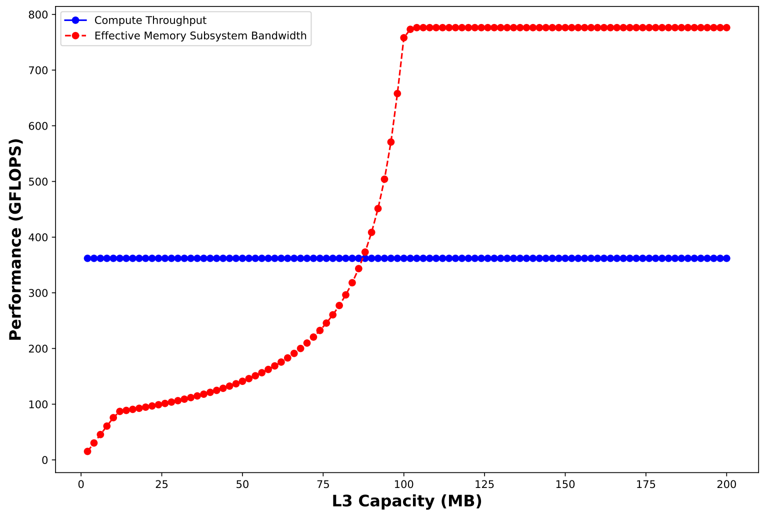

Figure 6.3: Line curve showing compute throughput and effective memory subsystem bandwidth versus L3 capacity for 4 channels of DDR5-4800 memory. The plot is drawn for application arithmetic intensity of 0.5 FLOPs/byte and working set size of 100 MB.

L3-main memory bandwidth. This is the case for L3 capacities from 12 MB to 88 MB. Further adding L3 slices increases L3 hit rate and effective L3-main memory bandwidth until the system performance becomes bounded by the compute throughput of the compute cores. This is true for L3 cache capacity greater than 88 MB. The relative positioning of the compute core-L3 bandwidth, effective L3-main memory bandwidth, effective memory subsystem bandwidth, and compute throughput curves change for different memory types, frequencies, and channel counts. In other words, for each memory configuration, the regions in which system performance is bounded by compute core-L3 bandwidth, effective L3-main memory bandwidth, and compute throughput differ. For instance, as shown in Figure 6.1, the system using 6 channels of DDR5-5600 memory has a smaller interval of L3 capacity during which the system performance is bounded by effective L3-main memory bandwidth since it has higher memory channel count and frequency.

6.2 Cost Composition

Next, we can understand the cost composition of each design point by examining the compute die, memory, interposer (for HBM systems), and package cost separately. We will compare the cost for systems with different memory configurations at iso-L3 capacity since the effect of adding an additional slice of L3 is the same across memory configurations. The effect of adding one additional slice of L3 is as follows. Compute die and interposer area (for HBM systems) increase marginally which lowers yield for compute die and interposer. It also decreases the number of compute dies and interposers that can be cut from a wafer. Consequently, compute die and interposer costs increase. Since a slightly larger L3 cache consumes more power, the area and cost of the package also increase marginally.

Figure 6.4 shows the compute die, memory, interposer, package, and total cost of systems with different memory configurations at iso-L3 capacity of 60 MB. We observe that the cost of compute die is greater for systems with six memory channels compared to four since the additional channels require additional memory controllers on the compute die, which takes up additional area on the wafer, reduces yield, and lowers the number of dies that can be harvested from a wafer. We see that memory cost also increases with the number of channels and frequency with HBM memory costing the most. Next, we consider the package cost. The package cost increases with the channel count since additional memory controllers require additional power and additional bumps on the package to transmit data to off-chip DRAM dies. The package

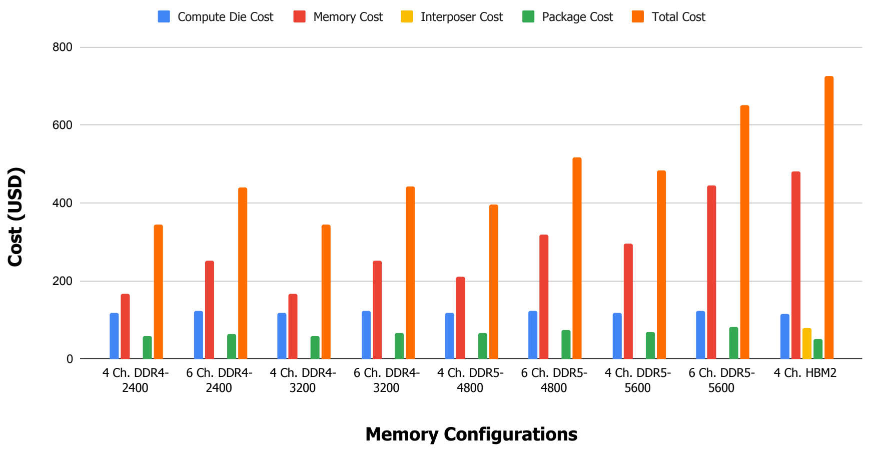

Figure 6.4: Compute die, memory, interposer, package, and total cost of systems with different memory configurations. All systems plotted at an L3 capacity of 60 MB.

cost also increases with frequency since the memory controller's operating voltage scales linearly with frequency. We observe that the system with HBM2 memory has the cheapest package. First, HBM's wide interface enables it to operate at a frequency lower than any DDR memories we considered while still achieving the highest bandwidth (we assume an operating frequency of 1000 MHz for HBM2). Second, since the HBM stack is on-chip compared to the off-chip DDR DIMM, the amount of energy required for the memory controller to signal a bit of data is approximately 4.29x @@(15pJ/b divide 3.5pJ/b) less. Both of these factors reduce the power consumption of the memory controller and thus the number of package bumps required for power delivery. Third, the integration of the DRAM stack within the package means memory interface wires connecting the HBM memory controller with the DRAM stack do not exit the package, which means fewer package-to-PCB bumps for signaling. These three factors outweigh the fact that the package for HBM systems needs to include bumps for power delivery to the HBM DRAM stack. For instance, compared to the 4-channel DDR4-3200 system, the 4-channel HBM2 system uses 26 fewer package bumps for power delivery to the memory controllers (due to lower memory frequency and energy per bit), 640 fewer package bumps for connections to off-chip DRAM DIMM, and 60 more bumps for power delivery to the HBM DRAM stack.

Overall, the system using HBM2 memory has the highest total cost with compute die, memory, interposer, and package accounting for 15.77%, 66.14%, 10.96%, and 7.14% of the total cost, respectively @@. The high cost of the HBM system relative to DDR systems is mostly attributed to (i) the inherently high cost of the HBM DRAM stacks which require the use of through-silicon vias and micro bumps compared to the widely available DRAM DIMMs and (ii) the high cost of interposer fabrication and assembly. In fact, in our HBM2 system with 60 MB of L3 cache, the interposer cost, including assembly, is approximately 69.49% of the compute die cost @@.

6.3 Cost-Performance Trade-off

Now that we have examined the performance behavior and cost composition of the design points, we can evaluate the cost and performance trade-off. Figure 6.5 shows overall system cost versus performance where each design point is represented by a marker on the plot. For a given memory configuration (represented as a unique combination of marker color and shape), the leftmost marker is the design point with 2 MB of L3 cache while the rightmost marker is the design point with 200 MB of L3 cache. The vertically stacked markers on the far right represent systems that are compute-bound, at which point adding additional L3 slices only increases cost without improving performance.

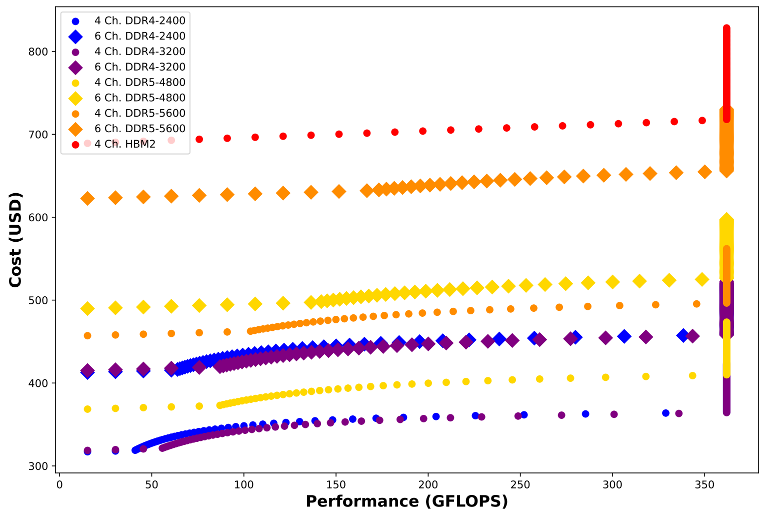

Figure 6.5: Scatter plot showing overall system cost versus performance. We show the data points for different memory types, frequencies, and channel counts. The plot is drawn for application arithmetic intensity of 0.5 FLOPs/byte and working set size of 100 MB.

From Figure 6.5, we can see that the same performance can be achieved using different combinations of L3 capacity, memory type, memory channel count, and memory frequency with very different system costs. For instance, Table 6.1 shows, for each memory configuration, the L3 capacity required to achieve 200 GFLOPS of performance (under this specific application profile of 0.5 FLOPs/byte arithmetic intensity and 100 MB working set) and the associated system cost. Therefore, the optimal system cost configuration is to use 4 channels of DDR4-3200 memory with 82 MB of L3 cache. Other system configurations may not have the best performance and cost trade-off but may be superior in other metrics such as area and power consumption. For each system configuration in Table 6.1, Figure 6.6 compares the compute die area, package area, and compute die power consumption (excluding power of HBM DRAM stack) normalized against the system with 4 channels of HBM2 memory and 26 MB of L3 cache. From Table 6.1 and Figure 6.6, we can conclude the 4-channel HBM2 system with 26 MB of L3 cache, despite having the highest system cost, outperforms all other DDR systems in Table 6.1 in compute die area, package area, and compute die power. In the iso-L3 capacity cost comparison in Section 6.2, we explained why HBM systems have small package areas and low compute die power. Another major contributing factor, in this case, is that the HBM system requires a much smaller L3 capacity to achieve 200 GFLOPs of performance, which lowers area and power consumption. This makes HBM a high-cost, low-procurability solution for systems with strict power and area constraints. Although Table 6.1 and Figure 6.6 are based on a target performance of 200 GFLOPS, the same conclusion (the HBM system has a higher system cost but lower compute die area, package area, and compute die power than any DDR system in

Table 6.1: L3 Capacity Required for 200 GFLOPS Performance and the Associated System Cost (Normalized against 4 Channel HBM2 System) for All Memory Configurations

Memory Configuration	**L3 Capacity (MB)**	Normalized System Cost
4 Ch. DDR4-2400	90	0.511
6 Ch. DDR4-2400	78	0.639
4 Ch. DDR4-3200	82	0.507
6 Ch. DDR4-3200	68	0.635
4 Ch. DDR5-4800	68	0.568
6 Ch. DDR5-4800	46	0.726
4 Ch. DDR5-5600	60	0.688
6 Ch. DDR5-5600	36	0.907
4 Ch. HBM2	26	1

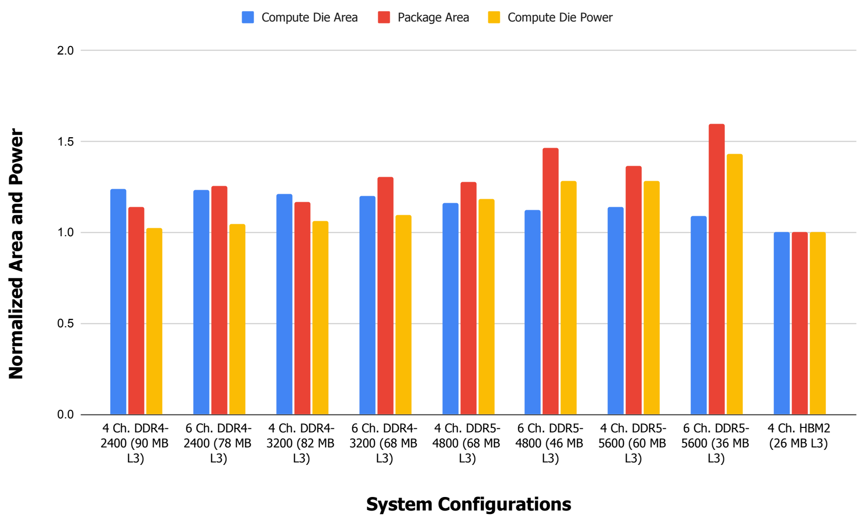

Figure 6.6: Compute die area, package area, and compute die power consumption (excluding power of HBM DRAM stack) for each design point shown in Table 6.1 normalized against the system with 4 channels of HBM2 memory and 26 MB of L3 cache.

iso-performance comparison), in fact, generalizes to all performance thresholds shown in Figure 6.5.

Since HBM systems have lower compute die power consumption than DDR systems for iso-performance comparisons, we analyzed the energy operational expenditure (OPEX) to understand if HBM systems have lower lifetime costs despite the higher upfront costs. For each system configuration in Table 6.1, Figure 6.7 and Figure 6.8 compares the system cost, energy OPEX, and lifetime cost with an assumed energy cost of 0.05 and 0.2 USD/kWh, respectively. We also assume five years of operation in the systems' lifetime. The trend of OPEX follows the trend of the compute die power consumption, as expected. The lifetime cost of the 4-channel HBM2 system is the third and sixth highest for energy costs of 0.05 and 0.2 USD/kWh, respectively. In other words, given the current cost of HBM DRAM stacks, DDR DIMMs, interposers, and electricity, DDR systems running at a low frequency with a large L3 cache still offer better economics over the system lifetime.

Since the attainable performance of a given system configuration depends significantly on the application profile, namely the arithmetic intensity and working set size, we next see if the same conclusion (the HBM system has a higher system cost but lower compute die area, package area, and compute die power than any DDR system in iso-performance comparison) generalizes

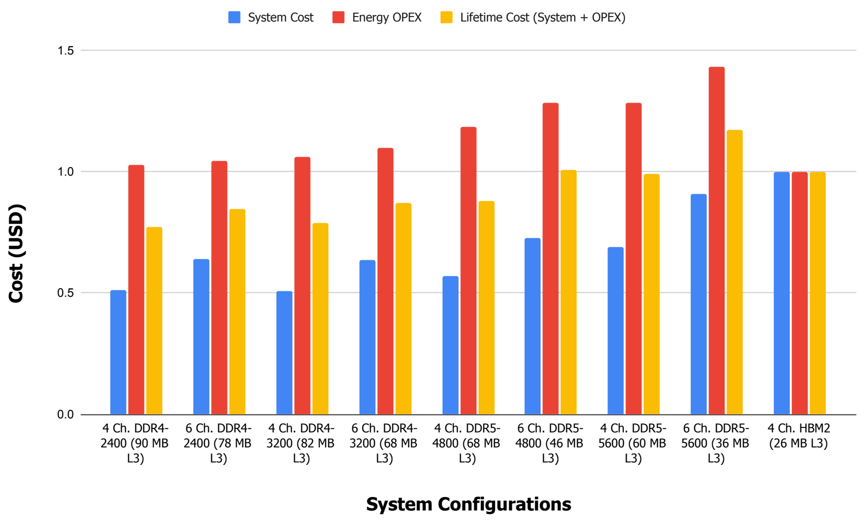

Figure 6.7: System cost, energy OPEX, and lifetime (system + OPEX) cost for each design point shown in Table 6.1 normalized against the system with 4 channels of HBM2 memory and 26 MB of L3 cache. Energy OPEX is estimated based on an energy cost of 0.05 USD/kWh and five years of operation. Only compute die power is considered.

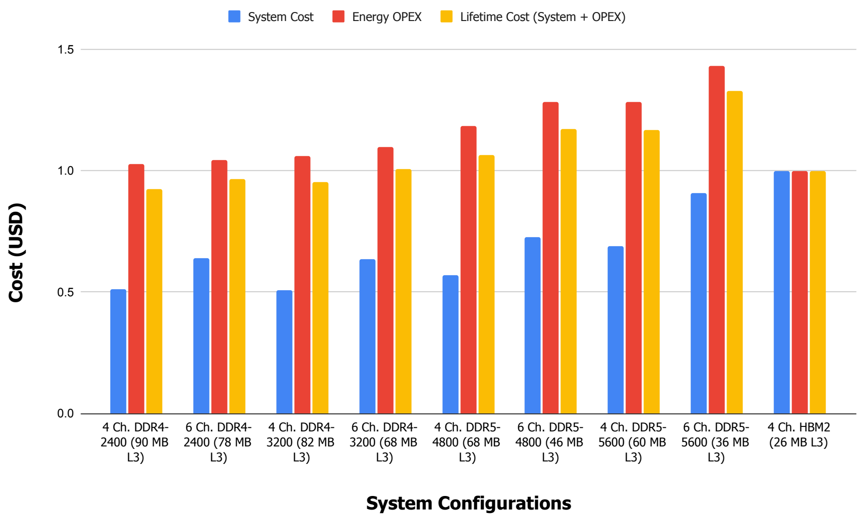

Figure 6.8: System cost, energy OPEX, and lifetime (system + OPEX) cost for each design point shown in Table 6.1 normalized against the system with 4 channels of HBM2 memory and 26 MB of L3 cache. Energy OPEX is estimated based on an energy cost of 0.2 USD/kWh and five years of operation. Only compute die power is considered.

to different application profiles. We reduce arithmetic intensity from 0.5 to 0.125 FLOPs/byte and increase the working set size from 100 MB to 150 MB to consider a more memory-intensive application profile. A larger working set implies a lower L3 hit rate and less filtering from L1 and L2 cache. Figure 6.9, similar to Figure 6.1, shows the system performance of design points versus L3 capacity. A line is drawn for a different combination of memory frequency

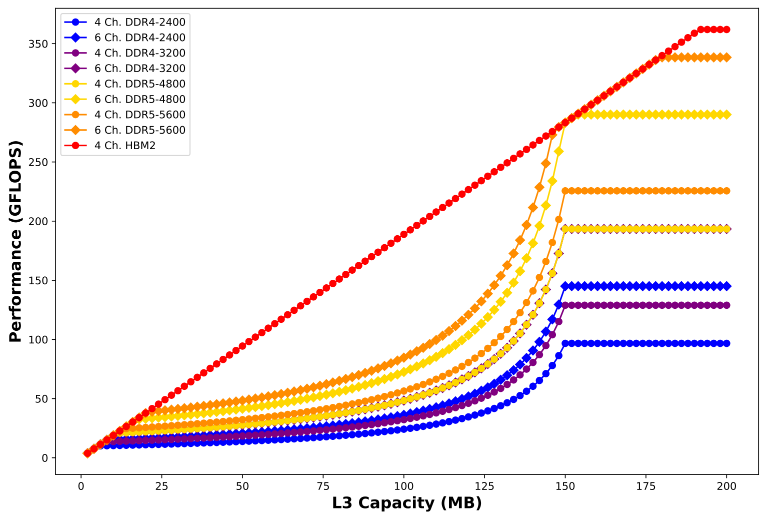

Figure 6.9: Line curve showing overall system performance versus L3 capacity. We show the curves for different memory types, frequencies, and channel counts. The plot is drawn for application arithmetic intensity of 0.125 FLOPs/byte and working set size of 150 MB.

and channel count. With a more memory-intensive application, only systems with 4 channels of HBM2 memory and more than 192 MB of L3 cache are able to fully utilize the compute throughput of the compute cores. For systems with any other memory configurations, the system performance eventually becomes bounded by the effective L3-main memory bandwidth. We also see alternating intervals of L3 capacity where the system performance is bounded by compute core-L3 bandwidth and effective L3-main memory bandwidth. For instance, the system with 6 channels DDR5-5600 memory is bounded by compute core-L3 bandwidth from 2 MB to 20 MB, bounded by effective L3-main memory bandwidth from 20 MB to 146 MB, bounded by compute core-L3 bandwidth from 146 MB to 178 MB, and finally bounded by effective L3-main memory bandwidth for L3 cache larger than 178 MB.

Figure 6.10, similar to Figure 6.5, shows overall system cost versus performance where each design point is represented by a marker on the plot. Similar to the case with 0.5 FLOPs/byte arithmetic intensity and 100 MB working set size, we observe that HBM systems have the highest system cost for all performance thresholds. Our model also shows that HBM systems have the lowest compute die area, package area, and compute die power for all performance thresholds compared to any systems using DDR memory. However, for the more memory-intensive

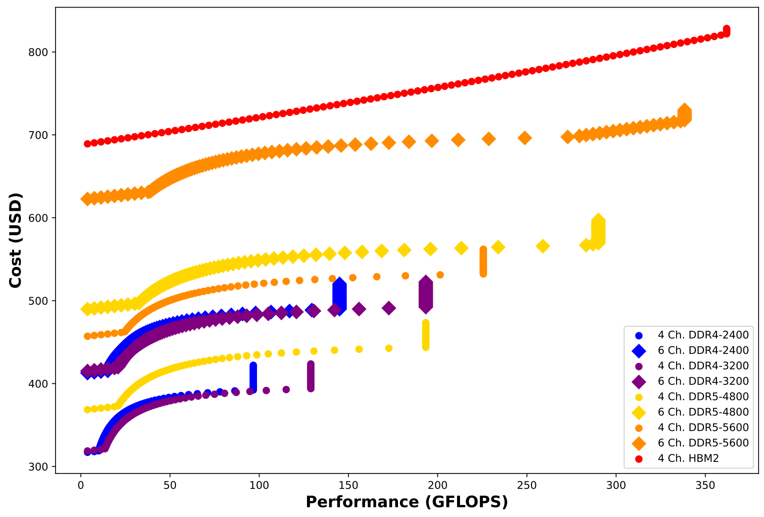

Figure 6.10: Scatter plot showing overall system cost versus performance. We show the data points for different memory types, frequencies, and channel counts. The plot is drawn for application arithmetic intensity of 0.125 FLOPs/byte and working set size of 150 MB.

application profile, the key difference is that there are performance levels not achievable using DDR memories regardless of channel count, frequency, or L3 capacity, namely performance greater than 338 GFLOPS.

For completeness, we consider a compute-intensive application profile with arithmetic intensity of 1 FLOPs/byte and a working set size of 50 MB. Figure 6.11, similar to Figures 6.1 and 6.9, shows the system performance of design points versus L3 capacity. We see that most memory configurations are compute-bound for most L3 capacities. Figure 6.12, similar to Figures 6.5 and 6.10, shows overall system cost versus performance where each design point is represented by a marker on the plot. We see the same conclusion holds that the HBM system has a higher system cost but lower compute die area, package area, and compute die power than any DDR system in iso-performance comparison (die area, package area, and compute die power scatter plots not shown).

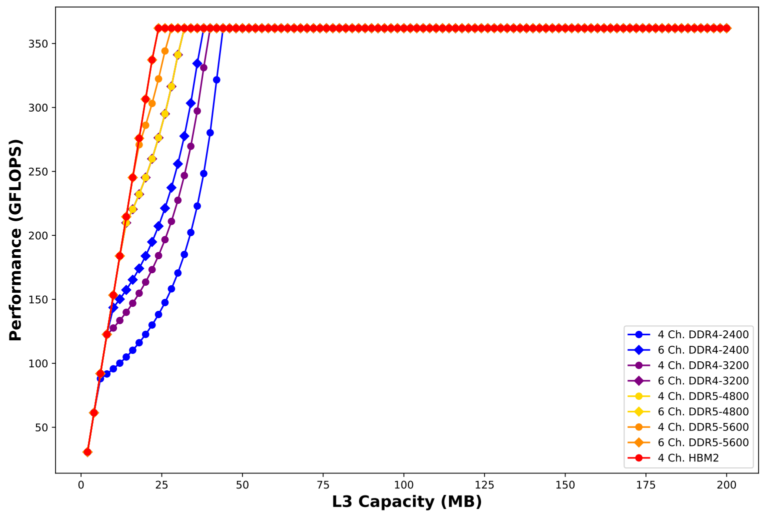

Figure 6.11: Line curve showing overall system performance versus L3 capacity. We show the curves for different memory types, frequencies, and channel counts. The plot is drawn for application arithmetic intensity of 1 FLOPs/byte and working set size of 50 MB.

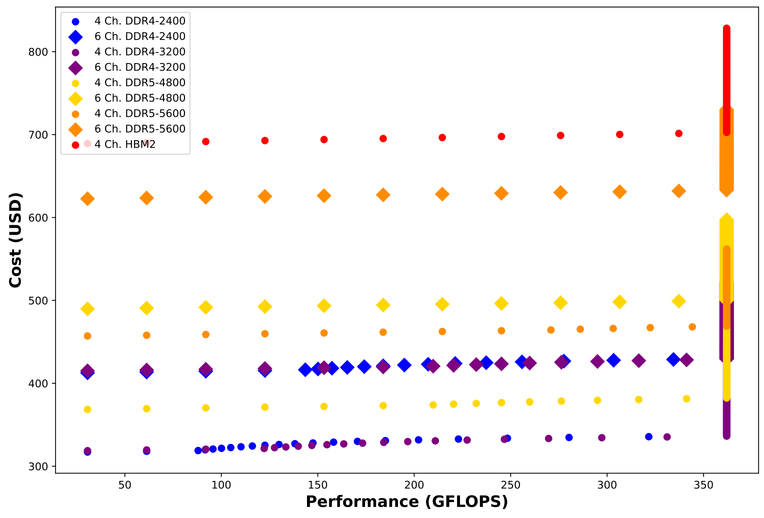

Figure 6.12: Scatter plot showing overall system cost versus performance. We show the data points for different memory types, frequencies, and channel counts. The plot is drawn for application arithmetic intensity of 1 FLOPs/byte and working set size of 50 MB.

Chapter 7

Conclusions

The unpredictable demand, complex supply chain, and high barrier of entry of the semiconductor industry make it prone to shortages. For companies from a wide range of industries relying on semiconductors for their products, designing with procurability and flexibility as a core architectural principle can be immensely valuable when shortages do occur. While HBM memory is a promising technology for applications requiring high DRAM bandwidth, it has lower procurability and higher cost compared to traditional off-chip DRAM DIMM due to the limited number of suppliers and the use of interposers.

In this study, applying our analytical model to design points with different L3 capacities, memory types, memory channel counts, and memory frequencies, we analyzed the performance characteristics of systems and identified regions where the performance is bounded by the compute core-L3 bandwidth, L3-main memory bandwidth, or compute throughput. On an iso-L3 capacity basis, we analyzed the cost composition of DDR versus HBM systems with HBM systems having the highest memory cost and total cost but the lowest package cost. Finally, combining performance and cost modeling, we identified design points where the same performance could be achieved at a lower cost by replacing HBM with DDR memory but with different combinations of a larger L3 cache, more memory channels, and higher memory frequency. Across a range of application profiles with different bandwidth requirements, we have shown that HBM systems have the highest system cost but the lowest compute die area, package area, and compute die power in iso-performance comparisons. OPEX analysis at different energy cost levels also shows that, despite the lower compute die power of the HBM systems, DDR systems operating at a low frequency with a large L3 cache still offer better lifetime costs.

Appendix A

Cache Organization Parameters

In this appendix, we list the cache organization parameters used as input to CACTI [16] in order to determine (i) the area of different levels of caches and (ii) the percentage of cache area dedicated to SRAM cells versus peripheral circuitry. These architectural parameters are based on the Intel Broadwell microarchitecture [25] with the cache capacity based on the Intel Xeon Platinum 8380 CPU used in our DSE [23].

Table A.1: Cache Organization Parameters for CACTI

Cache Level	**L1**	**L2**	**L3**
Capacity	64 kB	1 MB	5 MB*
Line Size	64 B	64 B	64 B
Associativity	8-way	8-way	16-way
Ports	2R + 2W	4R + 4W	2R + 2W
Tag/Data Array Access Order	Parallel	Hybrid	Sequential

* Although our DSE varies L3 capacity in a step size of 2 MB, we assume the L3 cache is formed from multiple 5 MB arrays in physical implementation. There are 2 read ports and 2 write ports for each 5 MB array. This approach will result in different L3 density and SRAM-cell-versus-peripheral-circuitry area percentages compared to a monolithic approach where the entire x MB L3 cache is made from a single SRAM array.

Bibliography

[1] D. Howley, "These 169 industries are being hit by the global chip shortage," *Yahoo Finance*, April 2021. [Online]. Available: finance.yahoo.com/news/these-industries-are-hit-hardest-by-the-global-chip-shortage-122854251.html

[2] U. B. of Economic Analysis, "Domestic auto production [daupsa]." [Online]. Available: fred.stlouisfed.org/series/DAUPSA

[3] M. Wayland, "Chip shortage expected to cost auto industry $210 billion in revenue in 2021," *CNBC*, September 2021. [Online]. Available: cnbc.com/2021/09/23/chip-shortage-expected-to-cost-auto-industry-210-billion-in-2021.html

[4] R. Kumar, "We're going to see another chip shortage— despite the chips and science act," *Fortune*, March 2023. [Online]. Available: fortune.com/2023/03/11/chips-and-science-act-semiconductor-shortage-rakesh-kumar/

[5] P. Hanbury, B. Radzevych, and B. Grant, "Engineering your way out of the global chip shortage," *Harvard Business Review*, December 2021. [Online]. Available: hbr.org/2021/12/engineering-your-way-out-of-the-global-chip-shortage

[6] JEDEC, "Jedec standard 212c.01: Graphics double data rate (gddr5) sgram standard," JEDEC, Tech. Rep., September 2022.

[7] ——, "Jedec standard 235: High bandwidth memory (hbm) dram," JEDEC, Tech. Rep., October 2013.

[8] D. Abril, “Computer monitors and other work-at-home essentials are in big demand in the coronavirus era,” *Fortune*, March 2020. [Online]. Available: fortune.com/2020/03/16/work-from-home-tech-products-sales/

[9] K. Lyons, “Us tightens trade restrictions on chinese chipmaker smic,” *The Verge*, September 2020. [Online]. Available: theverge.com/2020/9/26/21457350/us-tightens-trade-restrictions-china-chipmaker-smic

[10] “Crypto-miners are probably to blame for the graphics-chip shortage,” *The Economist*, June 2021. [Online]. Available: economist.com/graphic-detail/2021/06/19/crypto-miners-are-probably-to-blame-for-the-graphics-chip-shortage

[11] E. Barrett, “Taiwan’s drought is exposing just how much water chipmakers like tsmc use (and reuse),” *Fortune*, June 2021. [Online]. Available: fortune.com/2021/06/12/chip-shortage-taiwan-drought-tsmc-water-usage/

[12] S. K. Moore, “How and when the chip shortage will end, in 4 charts,” *IEEE Spectrum*, March 2023. [Online]. Available: spectrum.ieee.org/chip-shortage

[13] T. P. Morgan, “What faster and smarter hbm memory means for systems,” *The Next Platform*, July 2021. [Online]. Available: nextplatform.com/2021/07/21/what-faster-and-smarter-hbm-memory-means-for-systems/

[14] S. Pal, D. Petrisko, A. A. Bajwa, P. Gupta, S. S. Iyer, and R. Kumar, “A case for packageless processors,” in *2018 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, 2018, pp. 466–479.

[15] M. Mirza-Aghatabar, M. A. Breuer, S. K. Gupta, and S. Nazarian, “Theory of redundancy for logic circuits to maximize yield/area,” in *Thirteenth International Symposium on Quality Electronic Design (ISQED)*, 2012, pp. 663–671.

[16] S. Wilton and N. Jouppi, “Cacti: an enhanced cache access and cycle time model,” *IEEE Journal of Solid-State Circuits*, vol. 31, no. 5, pp. 677–688, 1996.

[17] S. Pal, D. Petrisko, R. Kumar, and P. Gupta, “Design space exploration for chiplet-assembly-based processors,” *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 28, no. 4, pp. 1062–1073, 2020.

[18] AnySilicon, “Die per wafer (free) calculator.” [Online]. Available: anysilicon.com/die-per-wafer-formula-free-calculators/

[19] Crucial, “Crucial 16gb ddr4-2400 udimm.” [Online]. Available: crucial.com/memory/ddr4/ct16g4dfd824a

[20] ——, “Crucial 16gb ddr4-3200 udimm.” [Online]. Available: crucial.com/memory/ddr4/ct16g4dfra32a

[21] ——, “Crucial 16gb ddr5-4800 udimm.” [Online]. Available: crucial.com/memory/ddr5/ct16g48c40u5

[22] ——, “Crucial 16gb ddr5-5600 udimm.” [Online]. Available: crucial.com/memory/ddr5/ct16g56c46u5

[23] Intel, “Intel xeon platinum 8380 processor.” [Online]. Available: intel.com/content/www/us/en/products/sku/212287/intel-xeon-platinum-8380-processor-60m-cache-2-30-ghz/specifications.html

[24] S. M. Khan and A. Mann, “Ai chips: What they are and why they matter,” *Center for Security and Emerging Technology*, pp. 44–45, April 2020.

[25] WikiChip, “Broadwell - microarchitectures - intel,” October 2014. [Online]. Available: en.wikichip.org/wiki/intel/microarchitectures/broadwell_(client)